Лекция 1. Локальные вычислительные сети (LAN)

Локальная вычислительная сеть (ЛВС, LAN – Local Area Network) – это совокупность аппаратного и программного обеспечения, позволяющего объединить компьютеры в единую распределенную систему обработки и хранения информации. К аппаратному обеспечению можно отнести компьютеры, с установленными на них сетевыми адаптерами, повторители, концентраторы, коммутаторы, мосты, маршрутизаторы и др., соединенные между собой сетевыми кабелями. К программному обеспечению можно отнести сетевые операционные системы и протоколы передачи информации. Расстояние между компьютерами объединяемыми в ЛВС обычно не превышает нескольких километров (термин "локальные сети"), что связано с затуханием электрического сигнала в кабелях. Технология виртуальных частных сетей (VPN – Virtual Private Network) позволяет через Internet или линии телефонной связи объединять в единую ЛВС несколько ЛВС, разнесенных на тысячи километров, однако это скорее именно объединение сетей и термин ЛВС здесь не достаточно точен.

Задачи, решаемые ЛВС

1. Передача файлов:
   Во-первых, экономится бумага и чернила принтера. Во-вторых, электрический сигнал по кабелю из отдела в отдел движется гораздо быстрее, чем любой сотрудник с документом. Он не болтает о футболе и не забывает в курилке важные документы. Кроме того, за электричество Вы платите гораздо меньше, чем зарплата курьера.
2. Разделение (совместное использование) файлов данных и программ:
   отпадает необходимость дублировать данные на каждом компьютере. В случае если данные бухгалтерии одновременно нужны дирекции, планово экономическому отделу и отделу маркетинга, то нет необходимости отнимать время и нервы у бухгалтера, отвлекая его от калькуляции себестоимости каждые три секунды. Кроме того, если бухгалтерию ведут несколько человек, то 20 независимых копий бухгалтерской программы и соответственно 20 копий главной книги (1 человек занимается зарплатой, 2-ой материалами и т.д.) создали бы большие трудности для совместной работы и невероятные трудности при попытке объединить все копии в одну. Сеть позволяет бухгалтерам работать с программой одновременно и видеть данные, вносимые друг другом.
3. Разделение (совместное использование) принтеров и другого оборудования:
   значительно экономятся средства на приобретение и ремонт техники, т.к. нет никакой необходимости устанавливать принтер у каждого компьютера, достаточно установить сетевой принтер.
4. Электронная почта:
   помимо экономии бумаги и оперативности доставки, исключается проблема "Был, но только что вышел. Зайдите (подождите) через полчаса", а также проблема "Мне не передали" и "А вы точно оставляли документы?". Когда бы занятый товарищ ни вернулся, письмо будет ждать его.
5. Координация совместной работы:
   при совместном решении задач, каждый может оставаться на рабочем месте, но работать "в команде". Для менеджера проекта значительно упрощается задача контроля и координирования действий, т.к. сеть создает единое, легко наблюдаемое виртуальное пространство с большой скоростью взаимодействия территориально разнесенных участников.
6. Упорядочивание делопроизводства, контроль доступа к информации, защита информации:
   Чем меньше потенциальных возможностей потерять (забыть, положить не в ту папку) документ, тем меньше таких случаев будет. В любом случае, гораздо легче найти документ на сервере (автоматический поиск, всегда известно авторство документа), чем в груде бумаг на столе. Сеть также позволяет проводить единую политику безопасности на предприятии, меньше полагаясь на сознательность сотрудников: всегда можно четко определить права доступа к документам и протоколировать все действия сотрудников.
7. Стиль и престиж:
   Играют не последнюю роль, особенно в высокотехнологичных областях.

1. Сетевые адаптеры (сетевые карты)

Существуют большое количество сетевых карт различных производителей, однако, по типу используемого протокола канального уровня, можно выделить три наиболее используемых типа:

1. Сетевая карта Ethernet (Fast Ethernet).

Самая распространенная сетевая карта. Используется в небольших офисных ЛВС и ЛВС среднего размера. Использование протокола Ethernet позволяет карте работать на скорости 10 Мбит/с, а протокола Fast Ethernet – 100 Мбит/с. Эти протоколы будут подробно рассмотрены далее в лекциях.

2. Сетевая карта Token Ring (High Speed Token Ring)

Сетевая карта для больших ЛВС. Использование протокола Token Ring позволяет карте работать на скоростях 4 и 16 Мбит/с, а протокола High Speed Token Ring – на скоростях 100 и 155 Мбит/с.

3. Сетевая карта FDDI (Fiber Distributed Data Interface)
Используется в оптоволоконных сетях. Протокол FDDI работает на скорости 100 Мбит/с и исторически, когда скорости других протоколов ограничивались 10-16 Мбит/с, из-за дороговизны оптоволоконных сетей использовался в основном на магистральных сетях передачи данных.

Выше были перечислены только самые распространенные протоколы, применяемые в сетевых картах. Существуют сетевые карты, поддерживающие и другие протоколы (например, 100VG-AnyLAN или GigabitEthernet). Подробнее об этих и других протоколах канального уровня будет сказано далее в лекциях. Существует также классификация сетевых карт, предложенная фирмой 3Com, делящая карты по четырем поколениям развития. Однако, поскольку эта классификация представляет больше исторический, чем практический интерес, то интересующихся отошлю к книге В.Г. Олифер, Н.А. Олифер "Компьютерные сети", с.274. Выпускаемые сейчас сетевые карты относятся в основном к четвертому поколению.

Сетевые карты также можно условно разделить на сетевые карты для клиентских компьютеров и сетевые карты для серверов. В сетевых картах для клиентских компьютеров значительная часть работы перекладывается на драйвер сетевой карты. Например, на стандартный драйвер фирм Microsoft и 3Com – драйвер NDIS (Network Driver Interface Specification), или драйвер ODI (Open Datalink Interface) фирмы Novell, использующийся в сетях NetWare. Благодаря такому подходу сетевая карта оказывается проще и дешевле, однако сильнее загружает центральный процессор компьютера, который вынужден выполнять часть функций сетевой карты, вместо выполнения прикладных задач пользователя. Поэтому сетевые карты, предназначенные для серверов, обычно снабжаются собственными процессорами, которые самостоятельно выполняют большую часть функций сетевой карты. Примером такой сетевой карты может служить сетевая карта SMS EtherPower со встроенным процессором Intel i960.

2. Сетевые кабели

Сетевые кабели бывают трех основных типов:
- витая пара (экранированная и неэкранированная)
- коаксиальный кабель (тонкий и толстый)
- оптоволоконный кабель (одномодовый, многомодовый).

Неэкранированная витая пара
Неэкранированная витая пара (UTP, unshielded twisted pair) - это кабель, в котором изолированная пара проводников скручена с небольшим числом витков на единицу длины. Скручивание проводников уменьшает электрические помехи извне при распространении сигналов по кабелю. Существует семь категорий витой пары:

Таблица

Категории кабеля витая пара

Категория	Характеристика
1-2	Устаревшие стандарты кабелей. Передача голоса и низкоскоростных данных (до 20 Кбит/с).
3	Наиболее широко распространенный на западе кабель телефонной проводки. Передача голоса и данных.
4	Улучшенный вариант категории 3. Повышенная помехоустойчивость и низкие потери сигнала. На практике используются редко.
5	Основной тип кабеля, используемый в современных компьютерных системах. Большинство новых высокоскоростных протоколов ориентируются именно на витую пару пятой категории.
6-7	Выпускаются сравнительно недавно. Основное назначение – поддержка высокоскоростных протоколов на отрезках кабеля большей длины, чем кабель категории 5. Кабель категории 7 по стоимости соизмерим с волоконно-оптическим кабелем, хотя характеристики волоконно-оптического кабеля выше. Поэтому ставится под сомнение целесообразность его применения.

В дальнейшем, если не будет отдельно оговорено, под витой парой будет пониматься витая пара пятой категории. Неэкранированная витая пара имеет волновое сопротивление 100 Ом (стандарт ISO 11801 допускает также 120 Ом). Неэкранированная витая пара подключается к сетевой карте через разъем, напоминающий телефонный. Внешне вид кабеля приведен ниже.



Рис. Неэкранированная витая пара

Кабель позволяет соединять напрямую только два компьютера, поэтому в сетях построенных на витой паре преобладает топология типа "звезда" (см. далее в лекциях), когда каждый из компьютеров, при помощи своего кабеля подключен напрямую к дополнительному сетевому устройству - концентратору (hub), который и обеспечивает взаимодействие между компьютерами в сети. Таким образом, при повреждении кабеля, сеть продолжит функционировать, а исчезнет связь только с одним компьютером, что легко диагностируется и устраняется. С другой стороны, при повреждении концентратора сеть станет недоступной для всех компьютеров, подключенных к нему.

Экранированная витая пара

В экранированной витой паре (STP, shielded twisted pair) изолированная пара проводников дополнительно помещена в экранирующую оплетку, что еще в большей степени увеличивает степень помехозащищенности сигналов. Экранированные витые пары внешне напоминают силовые электрокабели, используемые в быту.



Рис. Экранированная витая пара.

По экранированным витым парам передают только данные, голос не передают. Экранирование защищает передаваемые сигналы от внешних помех, а также уменьшает вредное для здоровья электромагнитное излучение. Однако наличие заземляемого экрана удорожает кабель и усложняет его прокладку. Кроме того, экранированная витая пара имеет волновое сопротивление 150 Ом, поэтому невозможно просто "улучшить" отдельные участки сети, путем замены неэкранированной витой пары (100 Ом) на экранированную – для этого потребуется также заменить сетевые адаптеры.

Тонкий коаксиальный кабель

Тонкий коаксиальный кабель RG-58 (иногда называется CheaperNet или ThinNet) представляет собой медный провод, экранированный при помощи оплетки. Толщина кабеля 6 мм. Волновое сопротивление 50 Ом. Схематично коаксиальный кабель изображен ниже. Следует отличать тонкий коаксиальный от телевизионного кабеля, применяемого в кабельном телевидении. Несмотря на схожесть, телевизионный кабель (RG-59) имеет волновое сопротивление 75 Ом и не предназначен для использования в компьютерной сети.



Рис. Коаксиальный кабель (схема).

Кабель к компьютерам в сети присоединяется при помощи Т-коннектора (см. рис.), оба конца кабеля должны заканчиваться терминаторами 50 Ом. При отсутствии терминатора в кабеле будут образовываться стоячие волны, что скорее всего приведет к неработоспособности всего сегмента сети. Схема подключения изображена ниже.

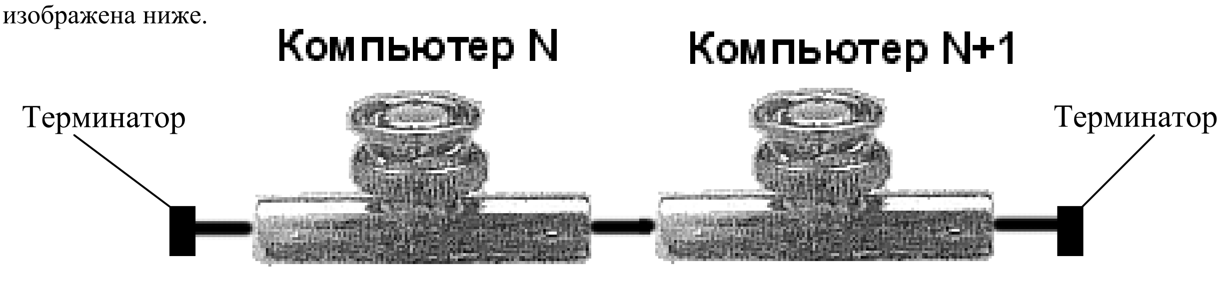


Рис. Подключение к тонкому коаксиальному кабелю при помощи Т-коннекторов (схема).

Сети, построенные на тонком кабеле Ethernet, имеют топологию "общая шина" (см. далее в лекциях), т.е. все компьютеры в сегменте сети подключены к одному кабелю. Из-за технических особенностей, при повреждении участка кабеля (или плохом контакте в Т-коннекторе или терминаторе) сеть не распадется на два изолированных, но работающих фрагмента, а полностью выходит из строя. Это снижает ее надежность, а также значительно затрудняет диагностику места возникновения неполадки. В связи с этим, перспективнее строить сети на основе кабеля "витая пара". Использование коаксиального кабеля считается устаревшей технологией, которая, например, даже не поддерживается протоколом Fast Ethernet. Тем не менее, большинство небольших офисных сетей продолжают использовать коаксиальный кабель, как исторически, так и просто экономя средства, т.к. использование "витой пары" предполагает покупку концентратора (hub) .

Толстый коаксиальный кабель
Толстый коаксиальный кабель (RG-8 и RG-11) имеет толщину 12 мм и бывает двух разновидностей: гибкий и жесткий. Он имеет большую степень помехозащищенности, большую механическую прочность, а также позволяет подключать новый компьютер к кабелю, не останавливая работу сети. Однако он сложен при прокладке, а для подключения к нему требуется специальное устройство (трансивер). Трансивер устанавливается непосредственно на кабеле контактно (прокалыванием) или бесконтактно, и питается от сетевого адаптера компьютера. Трансивер соединяется с сетевым адаптером при помощи кабеля AUI (Attachment Unit Interface) длиной до 50 метров. Сетевой адаптер должен иметь разъем AUI (разъем DB-15), который обычно имеется в концентраторах (hub-ах). Основная область применения толстого коаксиального кабеля – магистральные линии, соединяющие этажи здания (если использовать оптоволоконный кабель не позволяют средства).

Оптоволоконный кабель
В оптоволоконном кабеле для передачи сигналов используется свет. Он обычно состоит из центральной стеклянной нити толщиной в несколько микрон (световода), покрытой сплошной стеклянной оболочкой, обладающей меньшим показателем преломления, чем световод. Распространяясь по световоду, лучи света не выходят за его пределы, отражаясь от покрывающего слоя оболочки. Все это в свою очередь спрятано во внешнюю защитную оболочку. В первых оптоволоконных кабелях в качестве материала для световода использовалось стекло. В современных разработках используется также пластик. В качестве источников света в таких кабелях применяются светодиоды (длина волны 850 нм и 1300 нм) или полупроводниковые лазеры (длина волны 1300 нм и 1500 нм), а информация кодируется путем изменения интенсивности света. На приемном конце кабеля детектор преобразует световые импульсы в электрические сигналы. Волоконно-оптические кабели присоединяют к оборудованию разъемами MIC, ST и SC.
Различают следующие виды оптоволоконных кабелей:
- одномодовый кабель
- многомодовый кабель со ступенчатым изменением показателя преломления
- многомодовый кабель с плавным изменением показателя преломления

В одномодовом кабеле (Single Mode Fiber, SMF) используется центральный проводник очень малого диаметра, соизмеримого с длиной волны света — от 5 до 10 мкм. При этом практически все лучи света распространяются вдоль оптической оси световода, не отражаясь от внешнего проводника. В качестве источника света используется полупроводниковый лазер. Это самый дорогой тип кабеля, с самыми высокими показателями.

В многомодовых кабелях (Multi Mode Fiber, MMF) используются более широкие внутренние сердечники, которые легче изготовить технологически. В многомодовых кабелях во внутреннем проводнике одновременно существует несколько световых лучей, отражающихся от внешнего проводника под разными углами. Угол отражения луча называется модой луча. В качестве источников излучения в многомодовых кабелях применяются светодиоды, т.к. они дешевле. В целом, многомодовое волокно дешевле одномодового, хотя его характеристики хуже (больше затухание сигнала, уже полоса пропускания).

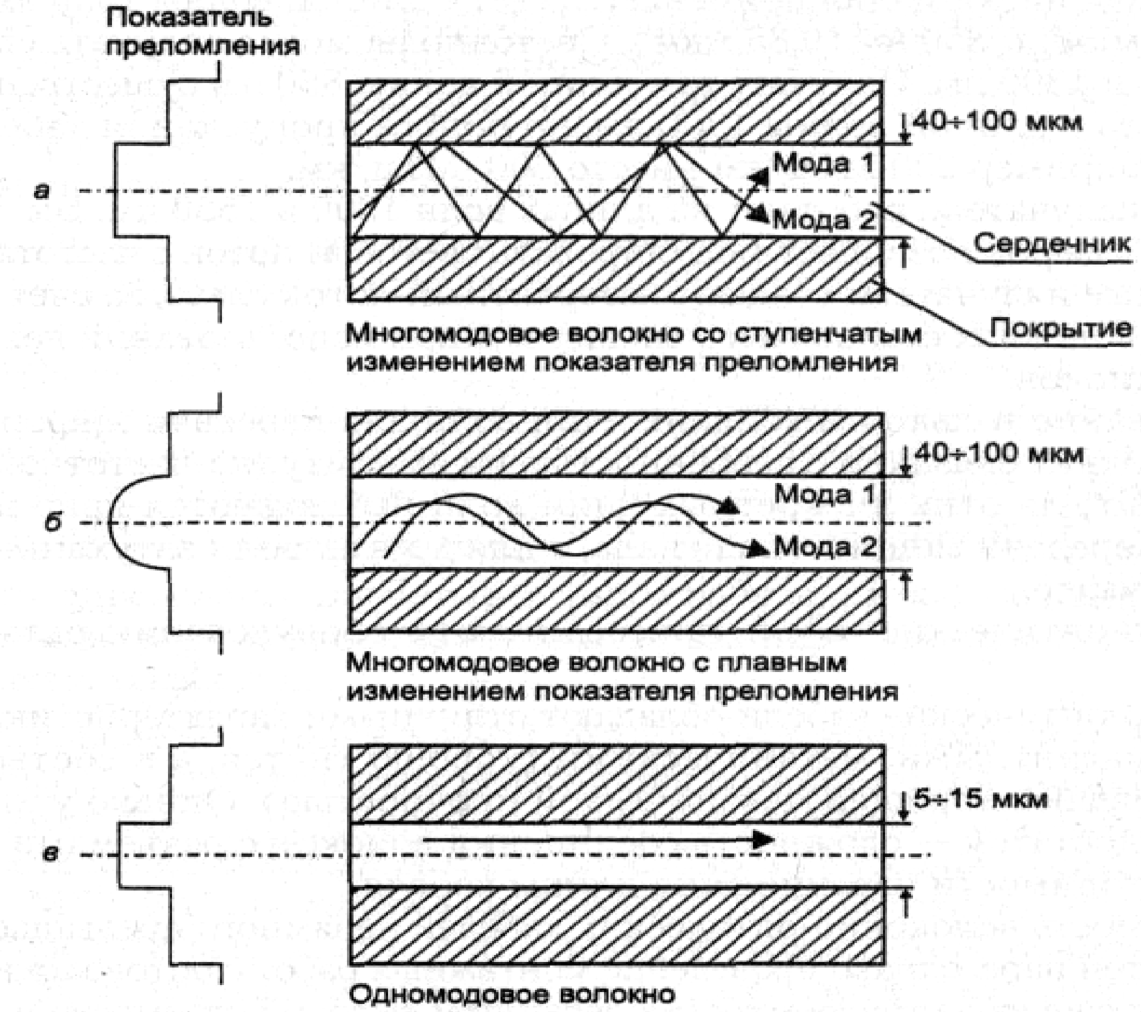
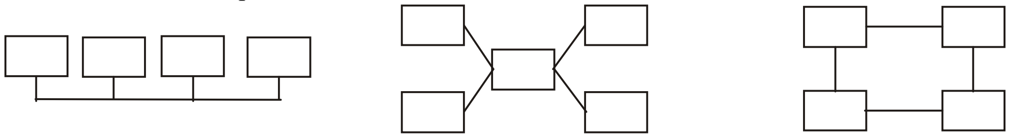


Рис. Типы оптоволоконного кабеля

Волоконно-оптические кабели обладают отличными характеристиками: защищенность от электромагнитных помех, механическая прочность (в изоляции) и хорошая гибкость. Однако у них есть серьезный недостаток — сложность соединения волокон с разъемами и между собой при необходимости наращивания длины кабеля. Сама стоимость волоконно-оптических кабелей ненамного превышает стоимость кабелей на витой паре, однако проведение монтажных работ с оптоволокном обходится намного дороже из-за трудоемкости операций и высокой стоимости применяемого монтажного оборудования. Так, присоединение оптического волокна к разъему требует проведения высокоточной обрезки волокна в плоскости строго перпендикулярной оси волокна, а также выполнения соединения путем сложной операции склеивания, а не обжатия, как это делается для витой пары. Выполнение же некачественных соединений сразу резко сужает полосу пропускания волоконно-оптических кабелей и линий. Для установки разъемов, создания ответвлений, поиска неисправностей в оптоволоконном кабеле необходима специальная аппаратура и высокая квалификация. Поэтому оптоволоконную линию чаще всего используют в качестве основной высокоскоростной магистрали крупной ЛВС, к которой через шлюзы (см. далее в лекциях) подключаются сегменты сетей отделов, построенные на "витой паре" или коаксиальном кабеле.

3. Топология сети

Под топологией подразумевается способ соединения компьютеров сетевыми кабелями. Самые распространенные топологии приведены ниже.



Общая шина   Звезда   Кольцевая топология

Рис. Наиболее распространенные топологии сетей.

Топология "общая шина" чаще всего используется для сетей Ethernet на коаксиальном кабеле, а топология "звезда" – для сетей на "витой паре". Кольцевая топология характерна для сетевых карт TokenRing, хотя кольцо там чисто логическое и физически сеть, скорее всего, будет построена по смешанной топологии "звезда" и "кольцо". В крупных ЛВС топология почти наверняка будет смешанной, а между компьютерами будет более одной связи. Возможны любые топологии, в том числе и не упомянутые выше.

4. Одноранговые сети и сети с выделенным файловым сервером.

Одноранговые сети – это сети равноправных компьютеров, т.е. каждый компьютер одновременно выполняет функции и рабочей станции (работают пользователи) и файлового сервера (хранение и разделение файлов). Для создания одноранговой сети в Windows 95 достаточно просто включить совместный доступ к файлам и принтерам (Пуск/Настройка/Сеть/Конфигурация/Доступ к файлам и принтерам) и создать сетевую папку, диск или принтер на своем компьютере, разрешив к ней доступ из сети (Мой компьютер/Выделить папку или диск/Контекстное меню/Доступ/Указать сетевое имя ресурса, установить разрешения доступа). Если это сделают и другие пользователи в сети, то вы сможете работать с дисками друг друга.

Преимущества одноранговой сети очевидны: экономятся деньги на покупке файлового сервера. Поэтому для маленьких ЛВС, где не требуется высокая производительность и надежность хранения данных и число пользователей невелико (2-11 человек), одноранговые сети являются эффективным решением. Однако у одноранговых сетей есть и серьезные недостатки:

1. Низкая скорость доступа к данным.
   В организации невозможно на каждом рабочем месте поставить высокопроизводительный компьютер. Поэтому, если к старенькому Pentium-у бухгалтера будут постоянно обращаться другие пользователи, то за компьютером работать станет просто невозможно, не говоря уже о том, что доступ к нему будет крайне замедлен.
2. Низкая надежность работы сети.
   По той же причине, что и в п.1, надежность старого Pentium-а и нового файлового сервера с системой дублирования дисков, источником бесперебойного питания, надежной файловой системой и т.д. просто несопоставимы. Кроме того, если бухгалтер, не озаботившись, что на его компьютер "пишут" другие пользователи, выдернет его из розетки, то данные могут быть просто испорчены.
3. Сложность администрирования сети.
   Построение профессиональной ЛВС подразумевает использование профессиональной сетевой операционной системы (типа Windows NT 5.0, NetWare или Linux). В одноранговой сети это сделать невозможно, т.к. рядовые пользователи не смогут, да и не должны разбираться в таких ОС. Кроме того, их компьютеры просто "не потянут" эти операционные системы из-за ограниченных возможностей аппаратуры. В связи с этим будет очень сложно управлять сетью, выставлять права доступа, вести журналы регистрации и т.д. Построение же защищенной сети станет вообще невозможным, т.к. Windows 95/98 – самая уязвимая, с точки зрения безопасности, операционная система.

В ЛВС с выделенным файловым сервером, компьютеры пользователей (рабочие станции) не разделяют диски друг друга. Вместо этого один (или несколько) компьютеров выделены исключительно для работы с файлами (файловый сервер), архивного хранения данных (сервер резервного копирования), управления печатью на сетевом принтере (сервер печати) или организации доступа в ЛВС по телефонным линиям (модемный пул). Пользователи за сервером не работают, за исключением редких случаев его настройки администратором сети, поэтому он может вообще не иметь монитора или иметь дешевый монохромный монитор. Для того, чтобы файл пользователя стал доступен другим людям в сети, он должен скопировать его на файловый сервер, в каталог, доступный другим пользователям. Преимущества построения сети с выделенным файловым серверам прямо противоположны недостаткам одноранговых сетей, однако это решение более дорогое.

Особенно хотелось бы остановиться на безопасности, которую может обеспечить такое построение сети. Во-первых, файл-сервер позволяет установить на нем профессиональную сетевую операционную систему с четкой моделью разграничения доступа, протоколированием доступа, надежными алгоритмами аутентификации (подтверждения личности пользователя) и шифрования. Так например, возможность шифрования файлов встроена вWindows 2000 Server уже на уровне операционной системы. Также сервер можно физически установить в хорошо защищенном и охраняемом помещении, а в качестве компьютеров пользователя использовать ***бездисковые рабочие станции,*** т.е. компьютеры, у которых отсутствует винчестер и дисковод, так что после выключения в них не сохраняется никакой информации (все файлы хранятся на сервере). В таком случае информацию нельзя несанкционированно скопировать на дискету, а кража бездисковой рабочей станции ничего не даст. Иногда бездисковые рабочие станции используют в целях экономии, однако это неверно, т.к. сильно повышает нагрузку на сеть. Кроме того, из-за специфического ПЗУ начальной загрузки, в будущем ее не удастся переделать в самостоятельный компьютер, даже установив винчестер.

Другим средством повышения защищенности сети, является организация удаленного доступа к ЛВС только через ***модемный пул***. Часто пользователи, сетевые компьютеры которых имеют модем,

настраивают их так, чтобы позвонив из дома можно было бы работать за компьютером также, как и сидя за ним в офисе. Однако, не являясь специалистами в области компьютерной безопасности, они тем самым создают "черный вход" для проникновения в ЛВС. Поэтому целесообразнее не предоставлять им такую возможность, а организовывать удаленный доступ в сеть только через специальный выделенный сервер, где будет вестись аутентификация (подтверждение личностей) пользователей, протоколирование их работы, а разрешения доступа будут выставлены с учетом того, что связь идет по телефонным линиям и может быть перехвачена.

5. Сетевое оборудование

Помимо серверов и рабочих станций в локальной сети используется большое количество дополнительного оборудования. В сети на витой паре, объединяющей больше двух компьютеров, будет, по крайней мере, один концентратор (hub), к которому и будут подключены все компьютеры. В более крупных сетях, скорее всего, появятся мосты, коммутаторы, маршрутизаторы. Подробнее все это оборудование будет рассмотрено в последующих лекциях, после рассмотрения сетевых протоколов.

6. Сетевые операционные системы.

Практически все современные ОС поддерживают работу в сети. Однако в качестве ОС для сервера чаще всего используются Nowell NetWare, Unix, Linux и Windows NT (Windows 2000 Server). Ниже будут кратко рассмотрены эти ОС, а также упомянуты такие ОС, как MacOS X, OS/2 Warp Server, BeOS и др. Не будут рассматриваться ОС MS Windows 95/98/ME, т.к. они представляют собой исключительно клиентские ОС, и построение сервера на базе этих операционных систем не рационально. Не будут также рассматриваться MS DOS 6.0 и MS Windows 3.11, т.к. они не отвечают современным требованиям.

6.1. ОС Nowell NetWare

Одна из первых коммерческих сетевых ОС, позволивших строить сети произвольной топологии, состоящих из разнородных компьютеров. Если раньше сетевые ОС сильно зависели от конкретной конфигурации сети, то ОС Nowell NetWare стала первой универсальной сетевой ОС. Любая сетевая карта, имеющая драйвер ODI (Open Datalink Interface) может использоваться в сетях Nowell. Благодаря такой универсальности ОС быстро завоевала рынок, и долгое время оставалась основной ОС для локальных сетей. С 1990 года даже фирма IBM стала перепродавать NetWare, и по сегодняшний день эта ОС используется достаточно широко.

Текущей версией ОС является NetWare 5.x. Помимо удобного графического интерфейса, эта версия NetWare имеет ряд других характерных особенностей:

1) NetWare 5.0 использует в качестве основного сетевого протокола TCP/IP (протокол, используемый в сети Internet). Если предыдущие версии NetWare работали на собственном протоколе фирмы Novell - протоколе IPX/SPX, а протокол TCP/IP мог использоваться только поверх IPX/SPX (также эмулировался NetBIOS), то теперь NetWare 5.0 предлагает следующие варианты:

- только протокол TCP/IP
- протокол TCP/IP в режиме "совместимости" (может использоваться IPX/SPX поверх TCP/IP)
- совместное использование протоколов TCP/IP и IPX/SPX (оба протокола работают параллельно и независимо)
- только протокол IPX/SPX.

2) В NetWare используется служба каталога NDS (Nowell Directory Service), которая представляет собой единую распределенную базу данных в виде дерева каталогов, в которой описываются все объекты сети (пользователи, группы пользователей, принтеры и т.д.), с указаниями прав доступа. База данных NDS является общей для всей сети. Если в предыдущих версиях NetWare 3.x и 2.x необходимо было создавать учетную запись пользователя (имя и пароль) на каждом сервере сети, то в NetWare 5.0 достаточно один раз зарегистрировать пользователя в NDS и он получит доступ ко всем серверам сети.

3) В NetWare используется мощная и гибкая модель разграничения доступа. Система безопасности подключения к сети включает в себя: ограничения на срок действия и частоту смены пароля, запрет на повторное использование старых паролей, ограничение времени суток и адресов компьютеров, с которых пользователь может подключаться к сети, запрет одному и тому же пользователю на подключение к сети с нескольких машин одновременно. Система безопасности файловой системы позволяет для каждого файла и каталога назначить различным пользователям любую комбинацию следующих прав доступа: чтение, запись, создание, удаление, модификация (имени файла и его атрибутов), просмотр (содержимого каталога), изменение прав доступа, супервизор (полный набор всех прав). Аналогично регулируется доступ и к любым другим объектам NDS (права на просмотр, создание, удаление, переименование объектов, чтение, запись, сравнение и добавление их свойств, права супервизора). NetWare имеет также двухстороннюю систему аудита: внешние независимые аудиторы могут анализировать события в сети, не имея доступа к секретным данным, в то же время, администраторы сети не имеют доступа к данным аудита.

4) В NetWare 5.0 поддерживаются как традиционные тома (аналог логических дисков), так и тома NSS (Novell Storage Services). Традиционные тома обеспечивают надежную файловую систему, основанную на обработке транзакций (при сбое, файлы восстанавливаются в состояние "до сбоя"), сжатие файлов и систему

зеркального отражения дисков (данные параллельно пишутся на два различных винчестера: при повреждении одного, информация будет считана с другого). Тома NSS могут иметь размер до 8 терабайт и хранить до 8 триллионов файлов. Доступ к томам NSS происходит гораздо быстрее, чем к традиционным томам. В качестве тома NSS может монтироваться CD-ROM и разделы DOS.
5) В NetWare 5.0 реализована распределенная система печати NDPS (Novell Distributed Print Services), которая была разработана совместно с компаниями Hewlett-Packard и Xerox и позволяет реализовать:

- двухсторонний обмен данными (компьютер имеет возможность передавать данные на принтер, и принтер имеет возможность передавать данные в компьютер).
- оповещение о событиях (принтер по сети имеет возможность оповестить технический персонал, например о том, что кончился тонер).
- автоматическая загрузка драйверов принтера, шрифтов и др. ресурсов на компьютеры, которым требуется производить распечатку документов.

6) В комплект поставки NetWare 5.0 входит мощный и простой в использовании Web-сервер FastTrack Server for NetWare, тесно интегрированный с NDS и поддерживающий большинство языков разработки приложений для Web. FastTrack Server призван заменить собой Novell Web Server, использовавшийся в предыдущих версиях NetWare.
7) В состав сервера NetWare 5.0 входит виртуальная машина Java, что позволяет запускать приложения и апплеты Java на сервере. Например, графическая утилита управления сервером ConsoleOne написана на языке Java.

К сожалению, объемы лекции не позволяют раскрыть всю гамму возможностей ОС NetWare (основной упор в последующих лекциях будет сделан на рассмотрении ОС Linux и Windows 2000 Server). Тем не менее, NetWare продолжает оставаться удобной и широко используемой сетевой ОС для сервера.

6.2. ОС Windows NT

Эта сетевая операционная система очень мощная и удобная в администрировании, т.к. имеет хорошо продуманный графический интерфейс, привычный пользователям Windows, и позволяющий автоматизировать и упростить выполнение типовых задач. Однако, с точки зрения сетевой безопасности, она оставляет желать лучшего. Негативную роль здесь играет и "закрытость" системы, т.е. отсутствие возможности изменить и протестировать ее программный код под свои нужды (как это возможно в FreeBSD или Linux). Если для быстрого развертывания и простоты обслуживания локальной сети целесообразно использовать Windows NT, то для Internet-сервера лучше использовать различные клоны Unix и Linux. Эти утверждения подтверждаются фактическим материалом: по данным исследовательского сайта **void.ru** только 16% серверов домена RU используют Windows NT, а 60% приходится на Linux и клон Unix ОС FreeBSD (остальные 24% либо не были протестированы в ходе исследований, либо приходятся на другие ОС, например, ОС Solaris – 2.5%). Последней версией Windows NT, в настоящее время является Windows NT 5.0 (Windows 2000 Server). В связи с важностью вопроса, эта операционная система будет рассмотрена в лекциях отдельно.

6.3. ОС Unix, Linux

ОС Unix является старейшей сетевой операционной системой (создана в 1969 г.) и по сегодняшний день использующейся в Internet (см. статистику выше). Существует множество клонов Unix – практически ничем не отличающихся друг от друга операционных систем разных производителей: FreeBSD, BSD Unix (университет Berkley), SunOS, Solaris (фирма Sun Microsystems), AIX (фирма IBM), HP-UX (фирмы Hewlet Packard), SCO (фирмы SCO) и др. Самым популярным клоном Unix пожалуй является FreeBSD, в основном из-за того, что ее исходные тексты распространяются свободно, что позволяет произвольно переделывать ОС "под себя", а также тестировать систему на отсутствие ошибок и "черного хода". В связи с этим, FreeBSD содержит гораздо меньше ошибок, чем коммерческие варианты Unix, т.к. отладкой и устранением ошибок занималась не одна компания, а все программистское сообщество.

К клонам Unix можно отнести и Linux, однако в последнее время он выделился в самостоятельную операционную систему и продолжает бурно развиваться. Существует множество дистрибутивов (пакетов установки) Linux различных фирм. Самые популярные из них – это Red Hat Linux (США) и Mandrake (Европа). Существуют также Slackware Linux, Corel Linux, Caldera OpenLinux, Debian Linux, SuSE Linux, Black Cat Linux, Connectiva Linux и др. Структура файловой системы, система разграничения доступа и основные команды в Linux и Unix сходны. С точки зрения пользователя, основным отличаем Linux от ранних версий Unix является удобный графический интерфейс, во многом сходный с интерфейсом Windows (особенно у графической рабочей среды Gnome), а основным преимуществом, по сравнению с Windows, - большая надежность и скорость работы, большая защищенность файловой системы (в том числе и от вирусов) и более профессиональные средства работы с локальной сетью и Internet. Для Linux существует и разрабатывается большое количество программного обеспечения: от офисного пакета Star Office и графического редактора Corel Draw, до мощных СУБД (DB2 фирмы IBM) и систем разработки программ на C++, Perl, Java и др. И хотя пока еще рано рекомендовать неопытному пользователю переходить на Linux (в основном из-за проблем с использованием русских шрифтов в приложениях – отсутствует единая прозрачная схема настройки), тем не менее, в будущем, Linux возможно займет значительное место в нише ОС для домашних компьютеров. Пока же, он четко удерживает статус "удобной ОС для профессионалов", а также

используется как ОС для устройств бытовой электроники. Подробнее ОС Linux будет рассмотрена далее в лекциях.

6.4. ОС MacOS X

Если традиционно фирма Apple создавала свою операционную систему MacOS, как удобную ОС для настольных компьютеров, то MacOS X ориентирована на использование в качестве ОС для сервера. Это операционная система для компьютеров Macintosh, в основу построения которой были положены те же принципы, что и в ОС Unix. Фирма Apple создала операционную систему, которая сочетает в себе удобный графический интерфейс MacOS и зарекомендовавший себя "профессионализм" Unix систем. Если ваш сервер является компьютером Macintosh (процессор PowerPC), то использование MacOS X в качестве ОС более чем приемлемо.

6.5. ОС OS/2 Warp Server 5

OS/2 Warp Server – это мощная серверная ОС, построенная на идеологии клиент-сервер (программа подразделяется на две части, которые работают совместно: одна – на компьютере клиента, вторая – на сервере). Первоначально OS/2 было совместной разработкой фирм IBM и Microsoft (поэтому в OS/2 поддерживалось программное обеспечение DOS и Windows). Однако впоследствии фирмы прекратили сотрудничество. IBM продолжила развитие OS/2, а вариант Microsoft, называвшийся OS/2 Lan Manager, в дальнейшем трансформировался в ОС Windows NT. В настоящее время, под OS/2 понимается вариант фирмы IBM. Помимо серверного варианта, существует и клиентский вариант OS/2 Warp Client 5.0.
Характерными чертами OS/2 является:

- высокая надежность работы.
- хорошо реализованная вытесняющая многозадачность (включая нити). Задачи подразделяются на приоритетные классы: критический, серверный, нормальный, отложенный. Внутри нормального класса приоритет формируется динамически.
- удобный объектно-ориентированный графический интерфейс, возможность работы в режиме командной строки, специальный язык REXX для написания командных файлов.
- простота конфигурирования практически любых настроек ОС.
- высокопроизводительная и надежная файловая система HPFS (High Perfomance File System). В серверной версии применяется "журналируемая ФС" (JFS), перенесенная из ОС AIX и обеспечивающая высокую надежность.
- мощная система разграничения доступа к данным, хорошая защищенность от вирусов и др.

6.6. ОС BeOS

BeOS - операционная система, которая была создана в 1996 году, в компании Be, Inc, как операционная система для мультимедийного интернет - компьютера BeBox. Однако этот компьютер не снискал большой популярности, и в дальнейшем BeOS была перенесена на платформы PowerPC (Macintosh) и Intel x86 (IBM).

BeOS - полностью графическая система, созданная специально для работы с мультимедиа и Internet. Система обладает удобным и приятным пользовательским интерфейсом. Графическая система BeOS реализована в клиент - серверной архитектуре, что открывает возможности для многопоточной обработки и разделения задач между процессорами. BeOS обладает хорошей поддержкой мультипроцессорности, надежностью и обладает средой разработки приложений на языке С++, что приводит к появлению множества бесплатных программ. BeOS способна работать с файлами в несколько террабайт. А ко всему этому BeOS распознает такие файловые системы как FAT16/32 (Windows), HFS (Mac OS), NFS и другие. Сама операционная система занимает меньше 200 Мб, хотя набор стандартного программного обеспечения намного шире, чем в Windows. А ставится BeOS не более 10 минут.
Хотя BeOS и не задумывалась, как серверная ОС, однако ее сетевые возможности очень широки:

- **Многопоточный сетевой доступ:** сетевые возможности BeOS максимально многопоточны, и используют для своей работы многопроцессорную обработку.
- **Использование TCP/IP:** сетевые возможности BeOS базируются на протоколе TCP/IP, "родном" протоколе Internet.
- **Разделение доступа к файлам, основанное на FTP:** встроенные в систему возможности распределения доступа к файлам базируются на протоколе FTP, что позволяет разделять доступ к файлам с пользователями, использующими BeOS, Windows, Unix, Mac OS или любую другую систему, поддерживающую протокол FTP.
- **Интегрированный Web-сервер:** Встроенные в систему простейшие возможности Web-сервера, позволяют публиковать web-страницы с первого дня использования системы. Эти возможности являются модульными и могут быть заменены более мощными сервисами сторонних производителей.
- **Интегрированный Web-броузер:** NetPositive, Web-броузер BeOS, предлагает быструю обработку Internet-страниц, или чтение документации по BeOS, прилагающейся на CD. NetPositive поддерживает спецификацию HTML 3.2, HTTP 1.0 и 1.1, FTP, SSL, и другие Internet-стандарты.

- **Интегрированные почтовые сервисы:** Почтовые сервисы Internet, включая почтового клиента POP3, встроены в BeOS. Эти возможности являются модульными, поэтому могут быть расширены или заменены сторонними разработчиками.
- **Интегрированный сервер Telnet: с**тандартный Internet-сервис Telnet встроен в систему.
- **Совместимость с Unix (Posix):** BeOS имеет полную функциональную совместимость с Unix, что делает возможным перекомпиляцию Unix-совместимого кода без каких-либо изменений. BeOS имеет Unix-подобную командную строку и оболочку bash. Эти возможности также могут быть задействованы удаленно с помощью Telnet.
- **Совместимость с сетями Microsoft:** BeOS включает в себя клиента для сетей Microsoft, позволяя BeOS-системам получать доступ к общим (shared) дискам, файловым серверам и доменам Windows.
- **Поддержка печати AppleTalk:** BeOS поддерживает печать на AppleTalk-базированные принтеры через стандартные сети Ethernet, позволяя печатать на стандартных лазерных принтерах AppleTalk. BeOS также поддерживает TCP/IP-базированные принтеры.

6.7. ОС QNX 6.0

ОС QNX была разработана канадской фирмой QNX Software Systems, Ltd. для систем реального времени, т.е. компьютерных систем в которых необходима "быстрая реакция" операционной системы (порядка нескольких микросекунд). Системы реального времени применяются в управлении технологическими процессами (автоматизированные производства, ТЭЦ, атомные станции).

Название QNX происходит от сокращения Quick Unix (быстрый Unix). QNX – это система, построенная по стандарту POSIX (общий стандарт для всех Unix - систем), но отличающаяся чрезвычайно небольшими размерами и быстродействием. Микроядро QNX занимает всего 32 килобайта. Этого удалось добиться за счет того, что в состав микроядра включены только самые необходимые функции (управление реальной памятью; создание, переключение и взаимодействие между процессами; управление сетевым взаимодействием), а все прочие менеджеры ресурсов ОС являются такими же процессами, как и процессы пользователей. ОС QNX не требовательна к аппаратуре: для ее нормальной работы достаточно достаточно Pentium 200 с 32 Мб RAM.

Несмотря на столь скромные размеры и требования QNX обеспечивает удобную графическую оболочку Photon, схожую с оболочкой Windows, а также оболочку X-Photon для поддержки приложений X-Windows. QNX позволяет запускать приложений Windows и DOS в режиме эмуляции, планируется возможность запускать Linux-программы (пока их перенос возможен только в виде исходных текстов). Файловая система QNX устойчива к внезапным отключениям питания. QNX также обеспечивает доступ к дискам с файловыми системами fat32 (Windows), ext2 (Linux) и ISO9660 (CD-ROM). В QNX имеется удобные графические средства визуальной разработки приложений, например Photon Application Builder.

Сетевые возможности QNX обширны и отражают специфику применения QNX в автоматизированных системах управления производством. QNX изначально проектировалась как сетевая операционная система. Сеть QNX напоминает скорее единую большую ЭВМ, чем просто набор персональных компьютеров. При использовании протокола QNET, сеть превращается в один виртуальный суперкомпьютер, создавая единый однородный набор ресурсов, доступ к которым возможен из любого места сети. QNET также предусматривает возможность одновременной работы по нескольким физически параллельным сетям (основная и резервная). Такое построение сети способно обеспечить надежность и гораздо более быструю реакцию системы, что важно в автоматизированных системах управления производством. В условиях производства используются программируемые контроллеры и другие устройства ввода/вывода, работающие в режиме реального времени, которые могут потребовать значительных ресурсов, для обработки получаемой от них информации. Сеть QNX позволяет сфокусировать вычислительную мощность системы на производственном оборудовании там, где это необходимо, не жертвуя в то же время интерфейсом пользователя. К дополнительным сетевым возможностям QNX также относятся:

- возможность динамического подключения и замены сетевых драйверов, изменение параметров сети без приостановки ее работы.
- одновременное параллельное сосуществование различных протоколов (например QNET и TCP/IP).
- возможность регулировки нагрузки сети "на лету" и автоматическая переконфигурация сети, при выходе узлов из строя.
- встроенный в ОС компактный веб-сервер Slinger, поддерживающий SSI и CGI, что позволяет выдавать динамическую информацию о состоянии техпроцесса в виде HTML-документов. Организация рабочего места оператора сводится к установке компьютера с ОС, поддерживающей TCP/IP и содержащей в себе Web-браузер (Internet Explorer, Netscape Navigator и т.д.). Можно также воспользоваться Web-браузером Voyager, входящий в комплект поставки QNX.
- поддержка IP – фильтров, позволяющих реализовывать межсетевые экраны, поддержка NAT.

В заключение хочется подчеркнуть, что основное назначение QNX – это использование в сетях автоматизированного управления производством реального времени. И хотя при помощи QNX можно организовать узел Internet или файл-сервер локальной сети, лучше для этих целей пользоваться другими операционными системами.

6.8. Операционные системы мейнфреймов (VSE/ESA, VM/ESA, OS/390)

Мэйнфреймы редко используются даже в ЛВС предприятий запада, и информация, приведенная ниже, предназначена только для ознакомления. Мэйнфреймы семейства ESA фирмы IBM (ES/9000 и System/390) представляют собой высокопроизводительные компьютеры с большим количеством ресурсов и высоким соотношением производительность/цена (при условии полной загрузки). Мэйнфреймы семейства ESA представляют собой эволюционное развитие ряда System/360 - System/370 и отличаются большим объемом возможностей, реализованных на аппаратном уровне: мультипроцессорную обработку, средства создания системных комплексов, объединяющих несколько мэйнфреймов, средства логического разделения ресурсов вычислительной системы, встроенный криптографический процессор, высокоэффективную архитектуру каналов ввода-вывода и т.д. Современные ОС для мейнфреймов ESA (VSE/ESA, VM/ESA, MVS/ESA) представляют собой развитие ОС, работавших на System/360, System/370.

1) ОС VSE/ESA (Virtual Storage Extended) ориентирована на использование в конечных и промежуточных узлах сетей. Она функционирует на наименее мощных моделях мэйнфреймов. VSE эффективно выполняет пакетную обработку и обработку транзакций в реальном времени. Основное же назначение VSE – поддержка ПО, разработанного еще для System/360.

2) ОС VM/ESA (Virtual Machine) - гибкая интерактивная ОС, поддерживающая одновременное функционирование большого числа различных ОС на одной вычислительной системе, благодаря механизму виртуальных машин (ВМ). Монитор виртуальных машин (МВМ) распределяет ресурсы между виртуальными машинами. У каждой ВМ создается "впечатление", что в ее монопольном распоряжении имеется целая ЭВМ со всеми ее ресурсами, и ВМ представляет собой самостоятельный компьютер. На самом же деле ВМ владеет не всеми ресурсами вычислительной системы, а лишь теми из них, которые для нее выделил МВМ. Причем, это может быть как часть реальных ресурсов, так и ресурс, которого в вычислительной системе на самом деле нет, но МВМ моделирует его для ВМ. Когда ВМ выполняет команду, происходит прерывание, управление передается МВМ, и он прозрачно для ВМ выполняет для нее эту команду на реальном оборудовании или моделирует ее выполнение. ОС VM/ESA в основном используется при разработке операционных систем, т.к. ошибка при отладке разрабатываемой операционной системы может привести к порче или "зависанию" виртуальной машины, но не отразится на работе всей вычислительной системы. ОС VM/ESA находит и промышленное применение.

3) ОС OS/390 в ранних версиях - MVS (Multiply Virtual Storage) - основная ОС для применения на наиболее мощных аппаратных средствах. Она обеспечивает наиболее эффективное управление ресурсами при пакетном и интерактивном режимах и обработке в реальном времени, возможно совмещение любых режимов. Обеспечивает также объединение вычислительных систем, динамическую реконфигурацию ввода-вывода, расширенные средства управления производительностью. OS/390 является стратегическим направлением в развитии ОС мейнфреймов. Все ОС ESA обладают набором средств анализа производительности и управления ею, но в OS/390 такой набор представлен наиболее полно.

Лекция 2. Принципы функционирования ЛВС: протоколы и адресация.

Протокол – это набор правил, в соответствии с которым компьютеры обмениваются информацией. Эти правила включают формат, время и последовательность передачи данных, способы контроля и коррекции ошибок. В соответствии с моделью OSI (Open System Interconnection) существует семь уровней протоколов:

1. Физический уровень

Побитовая передача сигналов в кабелях: типы кодирования и физические характеристики сигналов, скорость передачи сигналов и т.д.

2. Канальный уровень

Передача кадров данных между сетевыми картами компьютеров. В самом общем виде кадр данных – это группа битов, состоящая из заголовка кадра и поля данных. В заголовке указывается адрес отправителя, адрес получателя, контрольная сумма и т.п. Канальный уровень обеспечивает получение доступа к общей среде передачи данных, обнаружение ошибок в кадрах данных, их повторную передачу и др. Канальный уровень – это аппаратное взаимодействие сетевая карта – сетевая карта.

3. Сетевой уровень

Сетевая ***логическая*** адресация сетевая карта – сетевая карта. Если на канальном уровне MAC-адрес сетевой карты физически "зашит" в ней производителем и не может изменяться, то на сетевом уровне сетевой карте компьютера может быть назначен любой логический адрес. При замене сетевой карты, MAC-адрес новой карты неизбежно будет другим, однако логический адрес новой карты можно оставить прежним, не нарушая адресацию в сети. Сетевой уровень также позволяет использовать в одной сети сегменты, построенные на различных протоколах канального уровня (например, объединить в единую сеть сегмент на сетевых картах Ethernet и сегмент на сетевых картах Token Ring). Кроме того, сетевой уровень отвечает за маршрутизацию (доставку) пакетов данных вне зависимости от сложности топологии сети.

4. Транспортный уровень.

Обеспечивает надежность доставки пакетов данных: установка виртуального канала передачи данных между сетевыми картами, контроль искажения или утери пакетов данных, повторная передача пакетов данных при необходимости.

5. Сеансовый уровень.

На практике используется редко (чаще всего сеансовый и представительский уровни объединяют с прикладным уровнем). Сеансовый уровень управляет диалогом сетевая карта – сетевая карта: фиксирует, какая из сторон является активной в настоящий момент, предоставляет средства синхронизации, которые позволяют вставлять контрольные точки в длинные передачи данных, чтобы в случае сбоя можно было вернуться назад к последней контрольной точке, а не начинать все с начала.

6. Представительский уровень.

Позволяет менять форму представления информации, не меняя ее содержания. Например, преобразования кодировки ASCII в кодировку EBCDIC, или шифрование передаваемых по сети данных при помощи протокола SSL (Secure Socket Layer). При использовании SSL, с точки зрения прикладной программы ничего не меняется: взаимодействие между клиентом и сервером по сети происходит как обычно. Однако фактически, любые данные передаваемые программой в сеть, шифруются протоколом SSL на компьютере-отправителе, передаются по сети в зашифрованном виде, а затем дешифруются протоколом SSL на компьютере получателе, прозрачно (незаметно) для работающей сетевой программы.

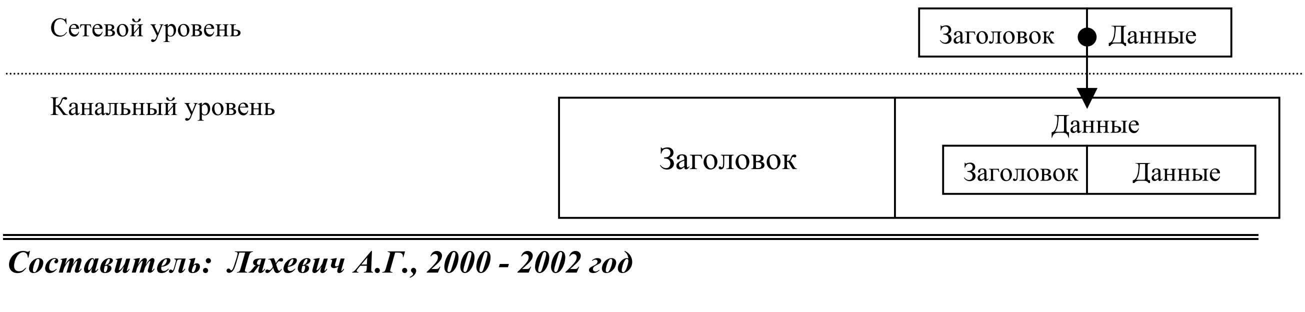
7. Прикладной уровень.

Набор разнообразных протоколов, при помощи которых взаимодействуют между собой прикладные программы. Каждая программа по желанию программиста может иметь свой собственный протокол или использовать один из широко-известных прикладных протоколов, например HTTP, SMTP, TELNET и др.

Модель OSI является международным стандартом, однако для практических целей, чаще всего пользуются упрощенной моделью в которой физический уровень подразумевается, но не рассматривается, а сеансовый и представительский уровни объединены с прикладным. Таким образом, упрощенная модель включает в себя:

- канальный уровень
- сетевой уровень
- транспортный уровень
- прикладной уровень

Важным понятием в многоуровневой модели протоколов является "инкапсуляция" пакетов. Чисто условно пакет можно представить в виде структуры [заголовок] – [данные]. В таком случае, инкапсуляцию можно представить следующей схемой:



При отправке, пакет сетевого уровня помещается в пакет (кадр) канального уровня, который и обеспечивает аппаратное взаимодействие сетевых карт, снимая эту задачу с протокола сетевого уровня. Протоколу сетевого уровня нет никакого дела до того, как реализован протокол канального уровня, а протокол канального уровня "не интересует" как работает протокол сетевого уровня – каждый выполняет свою часть работы. Инкапсуляция распространяется и на другие уровни: пакеты уровня приложения помещаются в пакеты транспортного уровня, которые в свою очередь помещаются в пакеты сетевого уровня, которые в свою очередь помещаются в пакеты канального уровня. Одним из следствий инкапсуляции является то, что при одном и том же протоколе канального уровня, может существовать несколько протоколов сетевого (транспортного, прикладного) уровня.

1. Протоколы канального уровня

1.1. Протокол Ethernet

Протокол Ethernet позволяет передавать данные со скоростью 10 Мбит/с и использовать следующие типы кабелей: толстый коаксиальный кабель (стандарт 10Base-5), тонкий коаксиал (стандарт 10Base-2), неэкранированную витую пару (стандарт 10Base-T), оптоволоконный кабель (стандарт 10Base-F).

Данные в протоколах канального уровня передаются в виде группы бит, организованных в кадр данных. Исторически существует 4 различных формата кадров Ethernet:

- кадр Ethernet DIX (Ethernet II) – один из первых форматов, стандарт фирм Digital, Intel и Xerox.
- кадр 802.3/LLC - международный стандарт.
- кадр Raw 802.3 (Novell 802.3) – стандарт фирмы Novell.
- кадр Ethernet SNAP – второй доработанный вариант международного стандарта.

Обычно сетевые карты автоматически распознают и поддерживают все четыре формата кадров. Для простоты изложения ограничимся рассмотрением самого простого по формату кадра Ethernet II, который имеет следующие поля:

Преамбула (для синхронизации) и признак начала кадра	Адрес назначения пакета	Адрес источника пакета	Тип пакета (указывает какому протоколу более высокого уровня принадлежит пакет)	Данные (передаваемая информация)	Контрольная сумма

Однако, помимо структуры кадра данных, в протоколе необходимо оговорить и порядок передачи этого кадра по сети. Основным принципом работы Ethernet является использование общей среды передачи данных разделяемой по времени, когда кадры данных передаются всеми компьютерами по общему кабелю. Особенно наглядно это проявляется при топологии "общая шина", хотя принцип сохраняется и при любой другой топологии. Впервые метод доступа к разделяемой общей среде был опробован во второй половине 60-х годов, в радиосети Aloha Гавайского университета, где общей средой передачи данных являлся радиоэфир. В 1975 году этот принцип был реализован и для коаксиального кабеля, в первой экспериментальной сети Ethernet Network фирмы Xerox.

В настоящее время сети Ethernet используется метод доступа CSMA/CD (Carrier Sense Multiply Access with Collision Detection) - коллективный доступ с проверкой несущей и обнаружением коллизий. Порядок передачи данных и коррекция ошибок происходит следующим образом: каждый кадр данных переданный в сеть получают все компьютеры, но только один из них распознает свой адрес и обрабатывает кадр. В каждый отдельный момент времени только один компьютер может передавать данные в сеть. Компьютер, который хочет передать кадр данных, прослушивает сеть и, если там отсутствует несущая частота (сигнал с частотой 5-10 Мгц), то он решает, что сеть свободна и начинает передавать кадр данных. Однако, может случится, что другой компьютер, не обнаружив несущей, тоже начнет передачу данных одновременно с первым. В таком случае, возникает столкновение (коллизия). Если один из передающих компьютеров обнаружил коллизию (передаваемый и наблюдаемый в кабеле сигнал отличаются), то он прекращает передачу кадра и усиливает ситуацию коллизии, посылкой в сеть специальных помех – последовательности из 32-бит (jam-последовательность), для того, чтобы и второй компьютер надежно обнаружил коллизию. После этого компьютеры ждут (каждый – случайное время) и повторяют передачу. Поскольку время – случайное (у каждого свое), то вероятность повторного столкновения невелика. Однако если столкновение произойдет снова (возможно с другими компьютерами), то следующий раз диапазон, в котором выбирается случайное время задержки, увеличится в 2 раза (после 10-й попытки увеличение не происходит, а после 16-й попытки кадр отбрасывается). В любом случае, время задержки, при возникновении коллизии невелико (максимум 52,4 миллисекунды) и незаметно для пользователя, однако при большой загрузке сети (начиная с 40 - 50%), слишком большая доля времени тратится на устранение коллизий и полезная пропускная способность падает. Более рациональным способом получения доступа к общей разделяемой среде является протокол Token Ring.

1.2. Протокол FastEthernet

Протокол Fast Ethernet был разработан совместными усилиями фирм SynOptics, 3Com (Fast Ethernet Alliance) и является развитием протокола Ethernet. Протокол Fast Ethernet позволяет передавать данные со скоростью 100 Мбит/с и использовать следующие типы кабелей: неэкранированную витую пару 5-й категории (стандарт 100Base-TX), неэкранированную витую пару 3-й категории (стандарт 100Base-T4), оптоволоконный кабель (стандарт 100Base-FX). Коаксиальный кабель в FastEthernet не поддерживается. Поддержка витой пары 3-й категории, несмотря на технические сложности, была реализована из-за того, что на западе, большинство уже проложенных телефонных кабелей, являются витой парой 3-й категории.

Метод доступа к разделяемой среде (CSMA/CD) в протоколе FastEternet остался прежним. Отличия от Ethernet заключаются в следующем:

- другой формат кадров
- другие временные параметры межкадрового и битового интервала (все параметры алгоритма доступа, измеренные в битовых интервалах сохранены прежними).
- признаком свободного состояния среды является передача по ней символа Idle (не занято), а не отсутствие сигнала, как в протоколе Ethernet.

Для совместимости со старыми сетевыми картами Ethernet, в протокол FastEthernet введена функция "автопереговоров" (auto-negotiation). При включении питания сетевой карты или по команде модуля управления сетевой карты начинается процесс "переговоров": сетевая карта посылает специальные служебные импульсы (FLP- fast link pulse burst), в которых предлагается самый приоритетный (с наибольшей скоростью передачи данных) протокол. Если второй компьютер поддерживает функцию "автопереговоров", то он ответит своими служебными импульсами, в которых согласится на предложенный протокол, или предложит другой (из поддерживаемых им). Если же на втором компьютере стоит старая сетевая карта Ethernet, не поддерживающая "автопереговоры", то ответа на запрос первого компьютера не последует, и он автоматически переключится на использование протокола Ethernet.

1.3. Протокол 100VG-AnyLan

Протокол 100VG-AnyLan был разработан совместными усилиями фирм Hewlett-Packard, AT&T и IBM.
И протокол FastEthernet и протокол 100VG-AnyLan являются развитием технологии Ethernet и позволяют работать на скорости 100 Мбит/с. Однако, если FastEthrnet ориентировался на минимальные изменения в протоколе Ethernet и совместимости со старыми сетевыми картами, то в протоколе 100VG-AnyLan, пользуясь сменой протоколов, была сделана попытка полностью отказаться от старых, и перейти к новым, более эффективным технологическим решениям.

Основным отличием 100VG-AnyLan является другой метод доступа к разделяемой среде - Demand Priority (приоритетный доступ по требованию), который обеспечивает более эффективное распределение пропускной способности сети, чем метод CSMA/CD. При доступе Demand Priority концентратору (hub-у) передаются функции арбитра, решающего проблему доступа к разделяемой среде. Сеть 100VG-AnyLAN состоит из центрального (корневого) концентратора, и соединенных с ним конечных узлов и других концентраторов (см. рис.). Допускаются три уровня каскадирования.

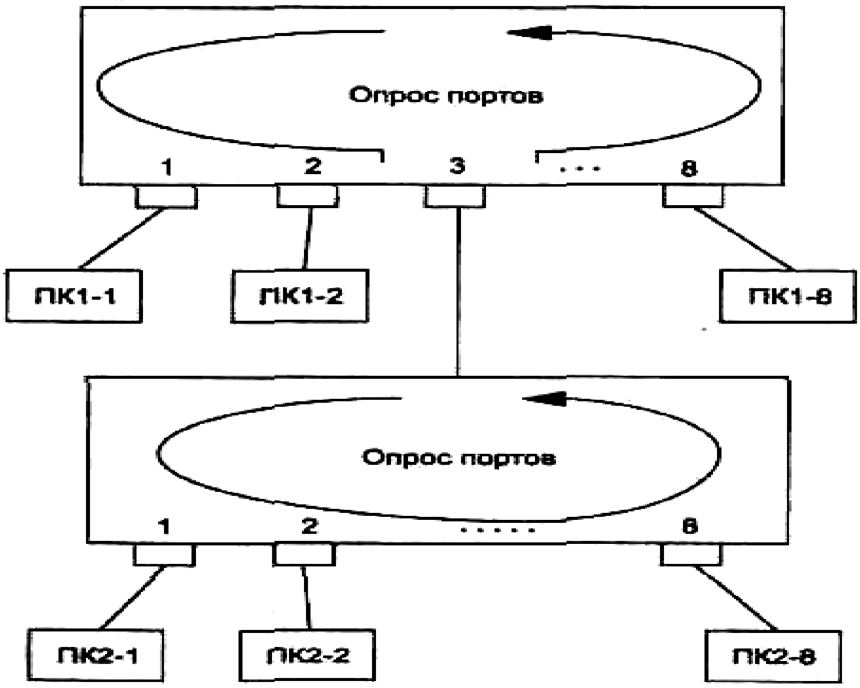


Рис.

Концентратор циклически выполняет опрос портов, к которым подключены компьютеры. Если к порту подключен другой концентратор, то опрос приостанавливается до завершения опроса концентратором нижнего уровня. Компьютер, желающий передать пакет, посылает специальный низкочастотный сигнал концентратору, запрашивая передачу кадра и указывая его приоритет: низкий (для обычных данных) или высокий (для данных, которые чувствительны к задержкам, например видеоизображение). Компьютер с низким уровнем приоритета, долго не имевший доступа к сети, получает высокий приоритет.

Если сеть свободна, то концентратор разрешает передачу пакета. Анализируется адрес назначения в пакете, и он передается на тот порт, к которому подключен соответствующий компьютер (адрес сетевой карты компьютера, подключенного к тому или иному порту, определяется автоматически, в момент физического подключения компьютера к концентратору). Если сеть занята, концентратор ставит полученный запрос в очередь. В очередь ставятся именно не сами кадры данных, а лишь запросы на их передачу. Запросы удовлетворяются в соответствии с порядком их поступления и с учетом приоритетов. У концентратора 100VG-AnyLan отсутствует внутренний буфера для хранения кадров, поэтому в каждый момент времени концентратор может принимать и передавать только один кадр данных – тот, до запроса на передачу которого дошла очередь (с учетом приоритетов).

В концентраторах 100VG-AnyLan поддерживаются кадры Ethernet и Token Ring (именно это обстоятельство дало добавку Any LAN в названии технологии). Каждый концентратор и сетевой адаптер 100VG-AnyLAN должен быть настроен либо на работу с кадрами Ethernet, либо с кадрами Token Ring, причем одновременно циркуляция обоих типов кадров не допускается. Другой особенностью является то, что кадры передаются не всем компьютерам сети, а только компьютеру назначения, что улучшает безопасность сети, т.к. кадры труднее перехватить при помощи анализаторов протоколов (снифферов).

Несмотря на много хороших технических решений, технология 100VG-AnyLAN не нашла большого количества сторонников и значительно уступает по популярности технологии Fast Ethernet.

1.4. Протокол GigabitEthernet

Протокол Gigabit Ethernet обеспечивает скорость передачи данных 1000 Мбит/с на всех основных типах кабельных систем: неэкранированная витая пара 5-ой категории, многомодовое и одномодовое оптоволокно (стандарты 1000Base-SX и 1000Base-LX), твинаксиальный кабель (коаксиальный кабель с двумя проводниками, каждый из которых помещен в экранирующую оплетку).

Протокол Gigabit Ethernet сохраняет максимально возможную преемственность с протоколами Ethernet и Fast Ethernet:

- сохраняются все форматы кадров Ethernet
- сохраняется метод доступа к разделяемой среде CSMA/CD. Поддерживается также полнодуплексный режим работы, когда данные передаются и принимаются одновременно (для отделения принимаемого сигнала от передаваемого сигнала, приемник вычитает из результирующего сигнала известный ему собственный сигнал).
- минимальный размер кадра увеличен (без учета преамбулы) с 64 до 512 байт. Для сокращения накладных расходов при использовании слишком длинных кадров для передачи небольших пакетов данных разработчики разрешили конечным узлам передавать несколько кадров подряд, без передачи среды другим станциям в режиме Burst Mode (монопольный пакетный режим). Если станции нужно передать несколько небольших пакетов данных, то она может не дополнять каждый кадр до размера в 512 байт (минимальный размер кадра), а передавать их подряд. Станция может передать подряд несколько кадров с общей длиной не более 65 536 бит или 8192 байт. Предел 8192 байт называется BurstLength. Если станция начала передавать кадр и предел BurstLength был достигнут в середине кадра, то кадр разрешается передать до конца.

1.5. Протокол Token Ring (High Speed Token Ring)

Использование протокола Token Ring позволяет карте работать на скоростях 4 и 16 Мбит/с, а протокола High Speed Token Ring – на скоростях 100 и 155 Мбит/с. Компания IBM является основным разработчиком протокола Token Ring, производя около 60 % сетевых адаптеров этой технологии.

Сеть Token Ring представляет собой кольцо: каждый компьютер соединен кабелем только с предыдущим и последующим компьютером в кольце. Физически это реализуется при помощи специальных концентраторов (см. рис.), которые обеспечивают целостность кольца даже при выключении или отказе одного из компьютеров, за счет обхода порта выключенного компьютера.

Принцип доступа к разделяемой среде – доступ с передачей маркера (token). Компьютер может начать передавать данные в сеть, только если получит от предыдущего компьютера в кольце "маркер" – специальный короткий пакет, свидетельствующий о том, что сеть свободна. Если компьютеру нечего передавать в сеть, то он передает маркер следующему компьютеру в кольце. Если компьютеру есть что передавать, то он уничтожает маркер и передает свой пакет в сеть. Пакет по битам ретранслируется по кольцу от компьютера к компьютеру, адресат получает пакет, устанавливает в пакете биты, подтверждающие, что пакет достиг адресата и передает пакет дальше по кольцу. Наконец, пакет возвращается к отправителю, который уничтожает его и передает в сеть новый маркер. Компьютер может и не передавать в сеть новый маркер, а продолжить передавать кадры данных до тех пор, пока не истечет время удержания

маркера (token holding time). После истечения времени удержания маркера компьютер обязан прекратить передачу собственных данных (текущий кадр разрешается завершить) и передать маркер далее по кольцу. Обычно время удержания маркера по умолчанию равно 10 мс.

В процессе работы сети, из-за сбоев, возможна потеря маркера. За наличие в сети маркера, причем единственной его копии, отвечает один из компьютеров - активный монитор. Если активный монитор не получает маркер в течение длительного времени (например 2,6 с), то он порождает новый маркер. Активный монитор выбирается во время инициализации кольца, как станция с максимальным значением MAC-адреса сетевой карты. Если активный монитор выходит из строя, процедура инициализации кольца повторяется и выбирается новый активный монитор. Чтобы сеть могла обнаружить отказ активного монитора, последний в работоспособном состоянии каждые 3 секунды генерирует специальный кадр своего присутствия. Если этот кадр не появляется в сети более 7 секунд, то остальные станции сети начинают процедуру выборов нового активного монитора.

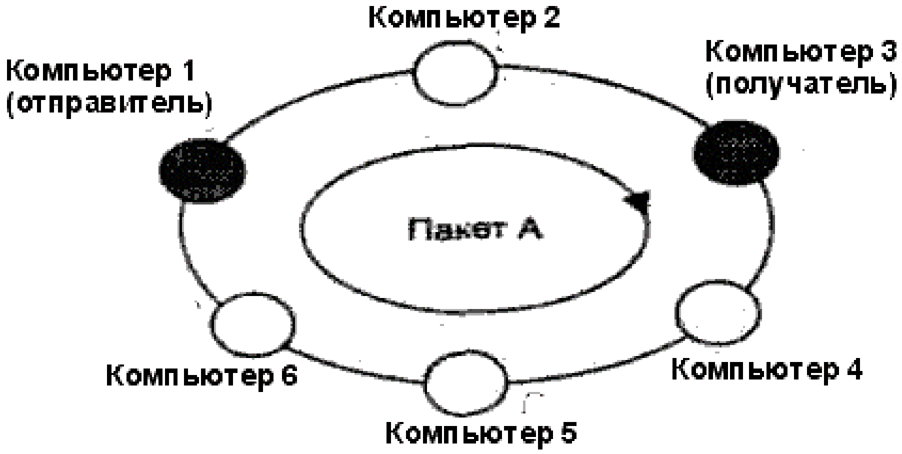


рис. Логическая структура сети Token Ring

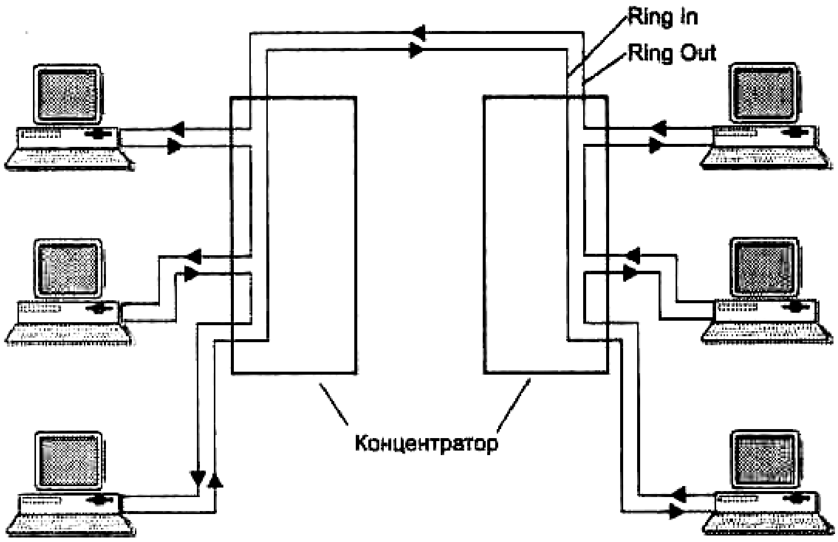


Рис. Физическая структура сети Token Ring

Описанный выше алгоритм доступа используется в сетях со скоростью 4 Мбит/с. В сетях со скорость 16 Мбит/с алгоритмы доступа более сложные: используется алгоритм доступа к кольцу, называемый алгоритмом раннего освобождения маркера (Early Token Release). Компьютер передает маркер доступа следующей станции сразу же после окончания передачи последнего бита кадра, не дожидаясь возвращения по кольцу этого кадра с битом подтверждения приема. В этом случае пропускная способность кольца используется более эффективно, так как по кольцу одновременно продвигаются кадры нескольких компьютеров. Тем не менее, свои кадры в каждый момент времени может генерировать только один компьютер — тот, который в данный момент владеет маркером доступа. Остальные компьютеры в это время только повторяют чужие кадры, так что принцип разделения кольца во времени сохраняется, ускоряется только процедура передачи владения кольцом.

Передаваемым кадрам, протокол верхнего уровня (например прикладного) может также назначить различные приоритеты: от 0 (низший) до 7 (высший). Маркер также всегда имеет некоторый уровень текущего приоритета и уровень резервного приоритета. При инициализации кольца основной и резервный приоритеты устанавливаются в ноль. Компьютер имеет право захватить переданный ему маркер только в том случае, если приоритет кадра, который он хочет передать, выше (или равен) текущему приоритету маркера. В противном случае компьютер обязан передать маркер следующему по кольцу компьютеру. Однако, даже если компьютер не захватил маркер, он может записать в поле резервного приоритета значение приоритета своего кадра (при условии, что предыдущие компьютеры не записали в это поле более высокий приоритет). При следующем обороте маркера резервный приоритет станет текущим и компьютер получит возможность захватить маркер.

Хотя механизм приоритетов в технологии Token Ring имеется, но он начинает работать только в том случае, когда приложение или прикладной протокол решают его использовать. Иначе все станции будут иметь равные права доступа к кольцу, что в основном и происходит на практике, так как большая часть приложений этим механизмом не пользуется.

Развитием протокола Token Ring стал протокол High-Speed Token Ring, который поддерживает скорости в 100 и 155 Мбит/с, сохраняя основные особенности технологии Token Ring 16 Мбит/с.

1.6. Протокол FDDI

Протокол FDDI (Fiber Distributed Data Interface) используется в оптоволоконных сетях и работает на скорости 100 Мбит/с. Исторически, когда скорости других протоколов ограничивались 10-16 Мбит/с, FDDI использовался на магистральных оптоволоконных сетях передачи данных.

Технология FDDI во многом основывается на технологии Token Ring, развивая и совершенствуя ее основные идеи. Сеть FDDI строится на основе ***двух*** оптоволоконных колец, которые образуют основной и резервный пути передачи данных между узлами сети. Наличие двух колец необходимо для повышения отказоустойчивости сети FDDI, и компьютеры, которые хотят воспользоваться этой повышенной надежностью могут (хотя это и не требуется) быть подключены к обоим кольцам.

В нормальном режиме работы сети данные проходят через все узлы и все участки кабеля только первичного (Primary) кольца. Этот режим назван режимом Thru — «сквозным» или «транзитным». Вторичное кольцо (Secondary) в этом режиме не используется. В случае какого-либо отказа, когда часть первичного кольца не может передавать данные (например, обрыв кабеля или отказ компьютера), первичное кольцо объединяется со вторичным (см. рис.), вновь образуя единое кольцо. Этот режим работы сети называется Wrap, то есть «свертывание» или «сворачивание» колец. Операция свертывания производится средствами концентраторов и/или сетевых карт FDDI. Для упрощения этой процедуры, данные по первичному кольцу всегда передаются в одном направлении, а по вторичному — в обратном (см. рис.). Поэтому при образовании общего кольца из двух колец, направление передачи данных по кольцам остается верным. Сеть FDDI может полностью восстанавливать свою работоспособность в случае единичных отказов ее элементов. При множественных отказах сеть распадается на несколько не связанных сетей.

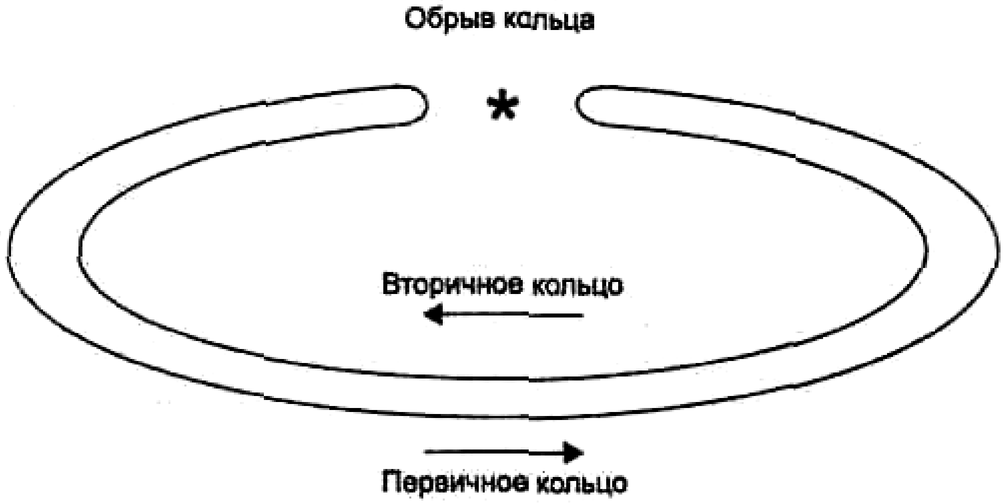


рис. Восстановление работоспособности сети FDDI при обрыве кольца.

Метод доступа к разделяемой среде в сети FDDI аналогичен методу доступа в сети Token Ring. Отличия заключаются в том, что время удержания маркера в сети FDDI не является постоянной величиной, как в

сети Token Ring, а зависит от загрузки кольца — при небольшой загрузке оно увеличивается, а при больших перегрузках может уменьшаться до нуля. В сети FDDI нет выделенного активного монитора — все компьютеры и концентраторы равноправны, и при обнаружении отклонений от нормы любой из них может начать процесс повторной инициализации сети, а затем и ее реконфигурации. В остальном пересылка кадров между станциями кольца полностью соответствует технологии Token Ring со скоростью 16 Мбит/с (применяется алгоритм раннего освобождения маркера).

На физическом уровне технология "сворачивания" колец реализуется специальными концентраторами. В стандарте FDDI допускаются два вида подсоединения компьютера к сети. Одновременное подключение к первичному и вторичному кольцам называется двойным подключением (Dual Attachment, DA). Компьютеры, подключенные таким образом, называются DAS (Dual Attachment Station), а концентраторы - DAC (Dual Attachment Concentrator). Подключение только к первичному кольцу называется одиночным подключением — Single Attachment, SA. Компьютеры, подключенные таким образом, называются SAS (Single Attachment Station), а концентраторы - SAC (Single Attachment Concentrator). Чтобы устройства легче было правильно присоединять к сети, их разъемы маркируются. Разъемы типа А и В должны быть у устройств с двойным подключением, разъем М (Master) имеется у концентратора для одиночного подключения станции, у которой ответный разъем должен иметь тип S (Slave). В случае однократного обрыва кабеля между устройствами с двойным подключением сеть FDDI сможет продолжить нормальную работу за счет автоматической реконфигурации внутренних путей передачи кадров между портами концентратора. При обрыве кабеля, идущего к компьютеру с одиночным подключением, он становится отрезанным от сети, а кольцо продолжает работать. Эта ситуация изображена на рисунке ниже.

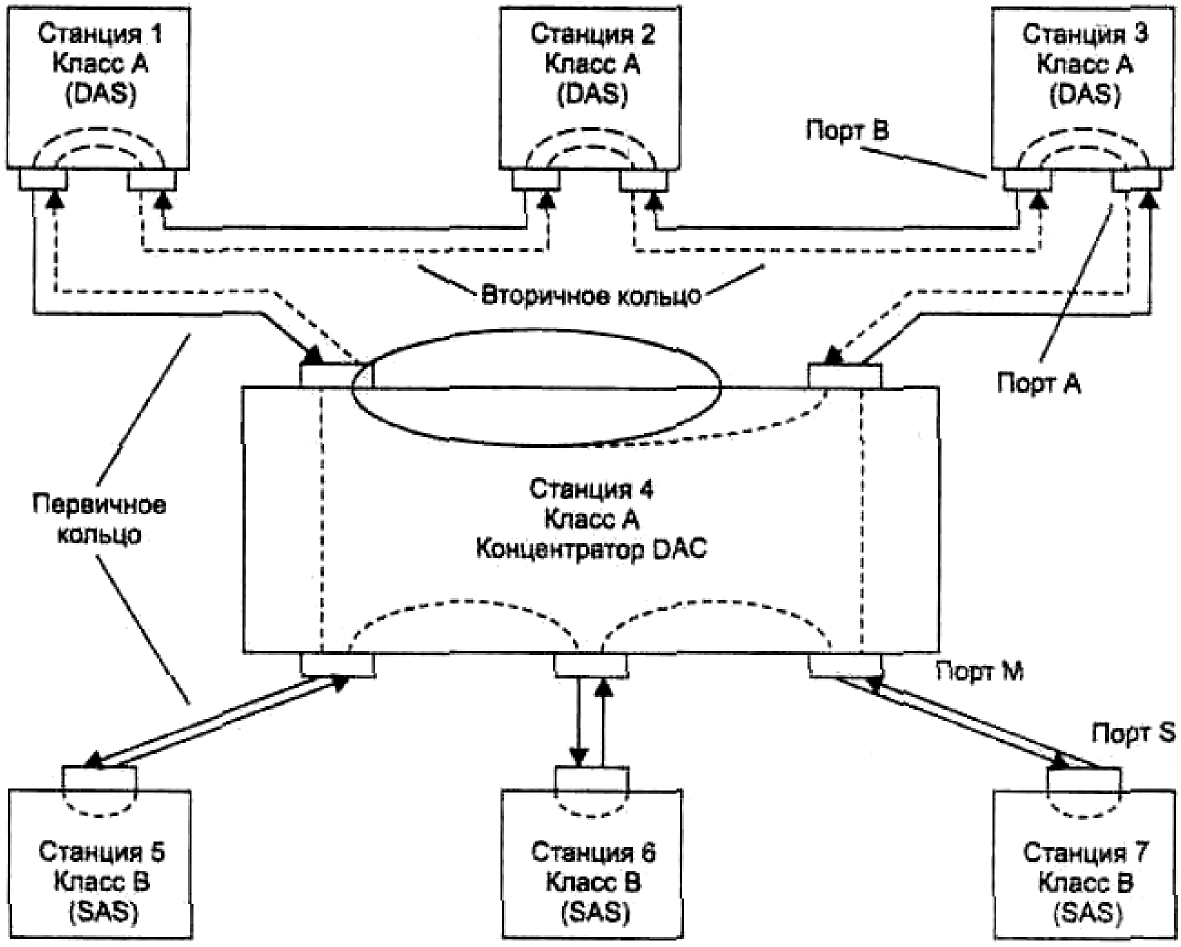


рис. Исходное подключение компьютеров к сети (до обрыва).

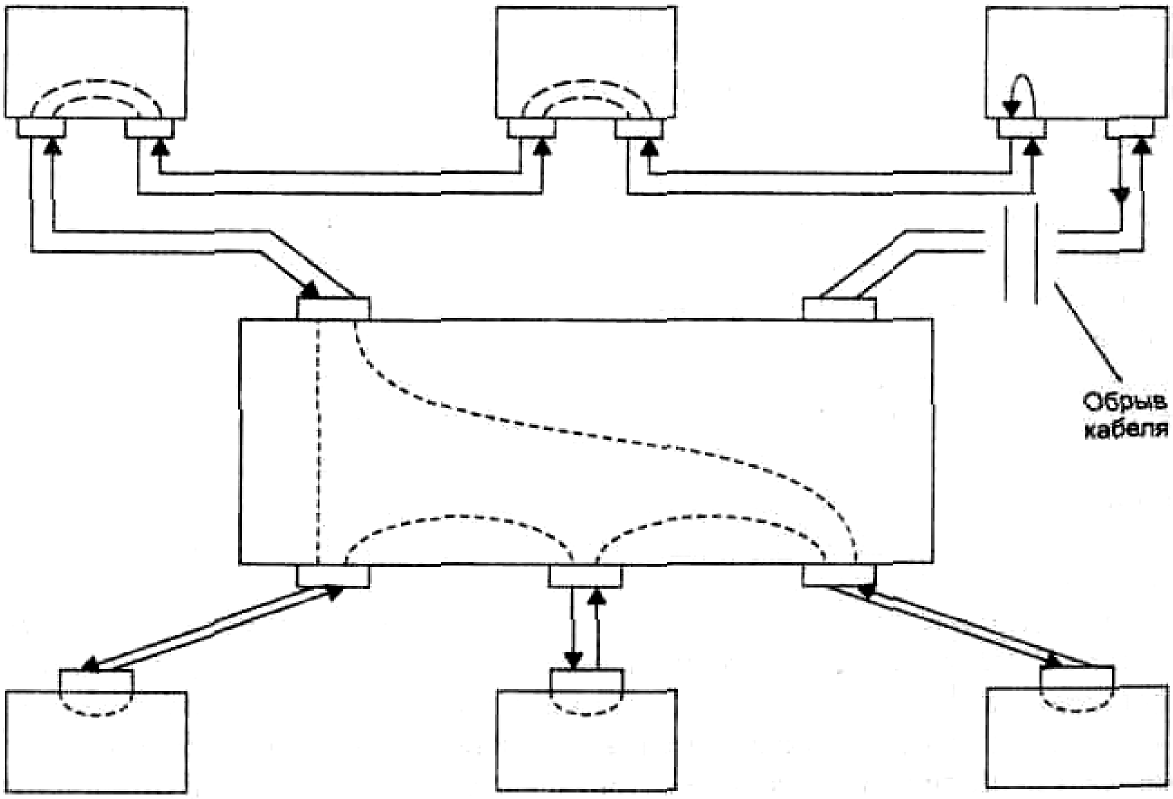


рис. Реконфигурация сети в случае обрыва.

1.7. Протоколы SLIP и PPP

Основное отличие протоколов SLIP и PPP от рассмотренных выше протоколов – это то, что они поддерживают связь "точка-точка", когда сетевой кабель используется для передачи информации только между двумя компьютерами (или другим сетевым оборудованием), соединенным этим кабелем. Такое соединение характерно при подключении к Internet по телефонной линии, при соединении локальных сетей между собой по выделенным или коммутируемым линиям, а также в сетях X.25, Frame Relay и ATM (см. далее в лекциях). Существует большое количество протоколов канального уровня для соединения "точка-точка", однако здесь мы ограничимся рассмотрением только SLIP и PPP.

SLIP (Serial Line IP) – протокол канального уровня, который позволяет использовать последовательную линию передачи данных (телефонную линию) для связи с другими компьютерами по протоколу IP (протокол сетевого уровня). SLIP появился достаточно давно, для связи между Unix – компьютерами по телефонным линиям и, в настоящее время, является устаревшим, т.к. не позволяет использовать протоколы сетевого уровня, отличные от IP, не позволяет согласовывать IP – адреса сторон и имеет слабую схему аутентификации (подтверждения личности) пользователя, заключающуюся в пересылке по сети имени и пароля пользователя. Таким образом, имя и пароль (даже зашифрованный) могут быть перехвачены и повторно использованы злоумышленником, или он может просто дождаться, пока пользователь пройдет аутентификацию, а затем отключить его и самому подключится от имени пользователя. Поэтому, большинство провайдеров Internet для подключения к своим машинам используют протокол PPP.

Протокол канального уровня PPP (Point to Point Protocol – протокол точка-точка) позволяет использовать не только протокол IP, но также и другие протоколы сетевого уровня (IPX, AppleTalk и др.). Достигается это за счет того, что в каждом кадре сообщения хранится не только 16-битная контрольная сумма, но и поле, задающее тип сетевого протокола. Протокол PPP также поддерживает сжатие заголовков IP-пакетов по методу Ван Джакобсона (VJ-сжатие), а также позволяет согласовать максимальный размер передаваемых дейтаграмм, IP-адреса сторон и др. Аутентификация в протоколе PPP является двусторонней, т.е. каждая из сторон может потребовать аутентификации другой. Процедура аутентификации проходит по одной из двух схем:

а) PAP (Password Authentication Protocol) – в начале соединения на сервер посылается имя пользователя и (возможно зашифрованный) пароль.

б) CHAP (Challenge Handshake Authentication Protocol) – в начале соединения сервер посылает клиенту случайный запрос (challenge). Клиент шифрует свой пароль, используя однонаправленную хэш-функцию (функция у которой по значению Y невозможно определить X) и запрос, в качестве ключа шифрования. Зашифрованный отклик (response) передается серверу, который, имея в своей базе данных пароль клиента, выполняет те же операции и, если полученный от клиента отклик совпадает с вычисленным сервером, то аутентификация считается успешной. Таким образом, пароль по линиям связи не передается. Даже если

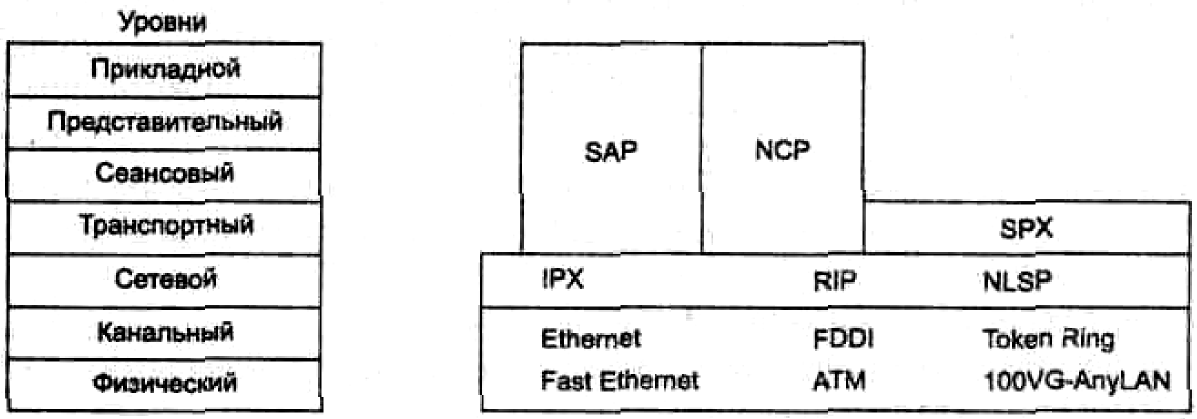
отклик клиента и будет перехвачен, то в следующий раз использовать его не удастся, т.к. запрос сервера будет другим. Определить же пароль на основании отклика – невозможно, т.к. хэш-функция шифрует данные только "в одну сторону". Для предотвращения вмешательства в соединение уже после прохождения клиентом аутентификации, в схеме CHAP сервер регулярно посылает испытательные запросы через равные промежутки времени. При отсутствии отклика или неверном отклике соединение прерывается.

2. Протоколы сетевого и транспортного уровня

Как и для канального уровня, существует несколько протоколов сетевого уровня. На практике, протокол сетевого уровня чаще всего разрабатывается и используется в паре с соответствующими протоколами транспортного, а иногда и прикладного уровня, образуя стек протоколов. В качестве примера можно привести следующие протоколы: IPX/SPX, NetBIOS/SMB, TCP/IP.

2.1. Стек протоколов IPX/SPX

Был разработан фирмой Novell для сетевой ОС NetWare, оптимизирован для использования в небольших локальных сетях, однако не удобен для глобальных сетей. Включает в себя протоколы IPX, SPX, SAP, NCP,



Протокол IPX (Internetwork Packet Exchange —межсетевой обмен пакетами) – протоколов сетевого уровня, поддерживает обмен пакетами (датаграммами) без установления канала связи и гарантии доставки пакета. Протокол IPX также отвечает за адресацию в сетях Net Ware. Адрес имеет формат: номер сети (задается администратором сети), адрес сетевой карты (определяется автоматически), номер сокета (идентифицирует приложение, пославшее пакет). Протокол IPX – аналог протокола IP из стека TCP/IP. Протокол IPX самый быстрый и экономит память, однако не дает гарантии доставки сообщения. За восстановлением утерянных или испорченных пакетов должен следить сам программист. Использование протокола SPX избавляет программиста от этой необходимости.

Протокол SPX (Sequenced Packet Exchange — последовательный обмен пакетами) – протокол транспортного уровня, поддерживает установление логического канала связи между компьютерами для обмена данными, коррекцию ошибок и, при необходимости, повторную передачу пакетов (аналог протокола TCP из стека TCP/IP).

Прикладной уровень стека IPX/SPX составляют два протокола: NCP и SAP. Протокол NCP (NetWare Core Protocol – протокол ядра NetWare) поддерживает все основные службы операционной системы Novell NetWare — файловую службу, службу печати и т. д. Протокол SAP (Service Advertising Protocol – протокол объявлений о сервисах) выполняет вспомогательную роль. С помощью протокола SAP каждый компьютер, который готов предоставить какую-либо службу для клиентов сети, объявляет об этом широковещательно по сети, указывая в SAP-пакетах тип службы (например, файловая), а также свой сетевой адрес. Наличие протокола SAP позволяет резко уменьшить административные работы по конфигурированию клиентского программного обеспечения, так как всю необходимую информацию для работы клиенты узнают из объявлений SAP.

Протоколы RIP (Routing Information Protocol) и NLSP (NetWare Link Service Protocol) отвечают за управление маршрутизацией (выбор маршрута доставки) пакетов, однако подробно здесь рассматриваться не будут. Протокол RIP реализован также и в стеке протоколов TCP/IP, а протокол NLSP аналогичен протоколу OSPF сетей TCP/IP.

2.2. Стек протоколов NetBIOS / SMB

Применяется фирмой Microsoft в своих сетевых ОС. В частности "сетевое окружение" работает при помощи этого протокола. NetBIOS включает в себя протоколы сетевого и транспортного уровня. Обеспечивает поддержку имен: каждая из рабочих станций в ЛВС может иметь одно или несколько имен (эти имена хранятся NetBIOS в таблице, в формате адрес сетевого адаптера – имя NetBIOS). Обеспечивает как обмен датаграммами, без установления канала связи и гарантии доставки сообщений, так и передачу пакетов с

установление логического канала связи между компьютерами с коррекцией ошибок и повторной передачей пакетов, при необходимости.

2.3. Стек протоколов TCP/IP

Протокол TCP/IP (Transmission Control Protocol/Internet Protocol – протокол контроля передачи данных / протокол передачи данных между сетями, Internet) разрабатывался Министерством Обороны США для глобальной сети ARPANET, и впоследствии стал основным протоколом, применяющимся в Internet. В состав стека протоколов TCP/IP входят протоколы: IP и ICMP – сетевой уровень, TCP и UDP – транспортный уровень. Ниже стек протоколов TCP/IP будет рассмотрено подробнее.

2.3.1. Протокол IP (ICMP)

Протокол IP отвечает за адресацию в сети и доставку пакетов между компьютерами сети, без установления соединения и гарантий доставки пакета. При использовании протокола IP, каждый компьютер в рамках сети должен иметь уникальный IP – адрес, представляющий собой 32-битное число. Для удобства чтения, IP адрес разбивают на четыре 8 битовых числа, называемых октетами, например 149.76.12.4. В локальной сети, которая не подключена к Internet или другим сетям, Вы можете назначать IP-адреса произвольно (главное, чтобы они не совпадали). Однако в Internet, IP-адреса выделяются централизовано, организацией InterNIC. InterNIC выдает адреса не на каждый отдельный компьютер, а в целом на локальную сеть.

В IP-адресе выделяют две части: сетевую часть (адрес локальной сети) и адрес компьютера в сети. Сетевая часть адреса может иметь переменную длину, которая зависит от класса IP-адреса и маски подсети. Выделяют следующие классы IP-адресов:

Класс А

включает сети с адресами от 1.0.0.0 до 127.0.0.0. Сетевой номер содержится в первом октете (1-127), что предусматривает 126 сетей по 1.6 миллионов компьютеров в каждой. Стандартная маска подсети для адреса класса имеет вид 255.0.0.0.

Класс В

Включает сети с адресами от 128.0.0.0 до 191.255.0.0. Сетевой номер находится в первых двух октетах (128.0 – 191.255), что предусматривает 16320 сетей с 65024 компьютерами в каждой. Стандартная маска подсети для адреса класса имеет вид 255.255.0.0.

Класс С

Включает сети с адресами от 192.0.0.0 до 223.255.255.0. Сетевой номер содержится в первых трех октетах (192.0.0 - 223.255.255). Это предполагает почти 2 миллиона сетей по 254 компьютеров в каждой. Стандартная маска подсети для адреса класса имеет вид 255.255.255.0.

Классы D

Включает адреса от 224.0.0.0 до 239.255.255.0. Эти адреса являются групповыми (multicast). Если нескольким компьютерам в сети назначен один и тот же групповой адрес, то пакет, адресованный на этот адрес, получат все компьютеры. Такие адреса в локальных сетях используются редко и зарезервированы для того времени, когда технические возможности сети Internet позволят организовывать теле- и радиовещание на группы компьютеров.

Классы E и F

Адреса попадающие в диапазон от 240.0.0.0 до 254.0.0.0 являются или экспериментальным, или сохранены для будущего использования и не определяют какую-либо сеть.

В примерах выше упоминалась "стандартная" маска подсети. Такая маска полностью соответствует классу адреса и может определяться автоматически, на основании анализа диапазона, в котором находится адрес. Казалось бы нет никакого смысла определять маску подсети вручную и вообще вводить такое понятие. Однако существуют ситуации, когда маска подсети будет отличаться от "стандартной". Допустим, у вас имеется сеть класса В (65024 компьютера) с IP-адресом 172.16.0.0 и вы хотите разбить ее на несколько подсетей, для разных филиалов предприятия. Стандартная маска подсети для адреса класса В равна 255.255.0.0 и адрес 172.16.1.0 интерпретируется, как компьютер с адресом 1.0 в сети с адресом 172.16. Однако если задать маску подсети равную 255.255.255.0, то этот IP-адрес прочитается как подсеть 172.16.1, содержащая 254 компьютера с адресами от 1 до 254. Таким образом, перед тем как решить является ли IP-адрес адресом конкретного компьютера или адресом сети, необходимо взглянуть на маску подсети, которая может отличаться от стандартной. Более того, маска подсети может не обязательно заканчиваться на границе байта. Маска всегда рассматривается в двоичном выражении, где единицы в октетах соответствуют полю адреса сети, а нули – полю адреса компьютера (см. табл.).

	Десятичное представление	Двоичное представление
IP-адрес	172 . 16 . 96 . 0	10101100 . 00010000 . 01100000 . 00000000
Маска подсети	255 . 255. 192 . 0	11111111 . 11111111 . 11000000 . 00000000
Интерпретация адреса: - адрес подсети - адрес компьютера	 172 . 16 . 1 32 . 0	 10101100 . 00010000 . 01 100000 . 00000000

Помимо адресов из классов A,B,C,D, E, F, ссуществует также несколько зарезервированных адресов. IP-адрес в котором все биты октеты адреса компьютера равны 0 относится ко всей сети, а где все биты октеты адреса компьютера равны 1 назван широковещательным (broadcast) адресом. Он относится к каждому компьютеру сети. Таким образом, 149.76.255.255 - не существующий адрес компьютера, который относится ко всем компьютерам из сети 149.76.0.0.

Имеются еще два зарезервированных IP-адреса, 0.0.0.0 и 127.0.0.0. Первый назван путь пакетов по умолчанию (default route), второй - кольцевым (loopback) адресом или ссылкой на самого себя. В несуществующей сети 127.0.0.0, адрес 127.0.0.1 будет назначен специальному интерфейсу, который действует подобно закрытому кругообороту. Любой IP пакет переданный на этот адрес будет возвращен на этот же компьютер так, как если бы пакет пришел откуда-то из сети. Это позволяет тестировать сетевое программное обеспечение без использования "реальной" сети.

Также имеется ряд "серых" IP-адресов, которые зарезервированы для использования только в локальных сетях. Пакеты с "серыми" адресами не передаются маршрутизаторами Internet. К таким адресам относятся:

Сеть класса A	10.0.0.0
Сеть класса B	от 172.16.0.0 до 172.31.0.0
Сеть класса C	от 192.168.0.0 до 192.168.255.0

По соображениям безопасности, рекомендуется использовать в локальных сетях только "серые" адреса. В таком случае прямой доступ из Internet к компьютерам ЛВС, в обход прокси-сервера организации, будет невозможен. При доставке, пакет от компьютера злоумышленника к компьютеру жертвы пройдет не один маршрутизатор Internet (алгоритмы маршрутизации см. ниже). Если адрес компьютера жертвы "серый", то первый же маршрутизатор Internet заблокирует пакет и не станет передавать его дальше. Таким образом, злоумышленнику придется сначала соединиться с прокси-сервером организации (на котором установлены средства аутентификации (проверки личности) пользователя, межсетевой экран и т.п.), и только прокси-сервер сможет обеспечивать контролируемое и протоколируемое взаимодействие между компьютером ЛВС и Internet, благодаря технологии NAT. Network Adress Translation (NAT) – это подмена в отправляемых и принимаемых пакетах данных "серых" IP-адресов компьютеров локальной сети на "реальный" IP-адрес прокси-сервера в сети Internet (более подробно см. далее в лекциях). Использование "серых" адресов также гарантирует, что даже если сообщение от одного компьютера ЛВС, к другом компьютеру ЛВС случайно попадет в каналы связи с Internet, то оно не будет передано дальше и не будет получено другой машиной, со случайно совпадающим IP-адресом.

Кроме адресации компьютеров в сети, протокол IP также отвечает за маршрутизацию (выбор маршрута доставки) пакетов данных в сетях с произвольной топологией. Маршрутизация происходит на основании специальных таблиц маршрутизации либо программно (сетевой операционной системой), либо при помощи специальных сетевых устройств – маршрутизаторов (подробнее маршрутизаторы будут рассмотрены далее в лекциях). Рассмотрим, как происходит доставка пакета по протоколу IP. В процессе рассмотрения будет частично затронут и протокол ARP (Address Resolution Protocol), позволяющий преобразовывать IP-адреса (сетевой уровень) в 6 байтные MAC-адреса сетевых карт Ethernet (канальный уровень):

1. Сеть состоит из отдельных сегментов (подсетей), которые соединены между собой либо маршрутизаторами, либо обычными компьютерами, на которых функции маршрутизации выполняются операционной системой. Такие компьютеры имеют несколько сетевых карт, каждая из которых имеет свой адрес в соответствующей подсети и являются шлюзами (gateway) из одной подсети в другую. Шлюзом называется любое сетевое оборудование с несколькими сетевыми интерфейсами и осуществляющее продвижение пакетов между сетями на уровне протоколов сетевого уровня.
2. Адресация в сетях идет по протоколу IP, поэтому компьютер-отправитель знает IP-адрес получателя. Но для доставки пакета на аппаратном уровне необходимо знать Ethernet-адрес сетевой карты получателя. Для этого по протоколу ARP посылается широковещательное сообщение всем компьютерам в данном сегменте сети. Все компьютеры получают его, но только компьютер с указанным IP-адресом "отзывается" и сообщает Ethernet-адрес своей сетевой карты. Компьютер отправитель кэширует ответ в своей памяти и в дальнейшем (пока кэш не будет очищен) будет направлять пакеты по этому Etehrnet-адресу. Таким образом, доставка в рамках одного сегмента сети происходит напрямую.
3. Однако компьютер-получатель может и не находится в одном сегменте с отправителем (что видно по маске подсети). В таком случае, сообщение будет послано на маршрутизатор (IP-адрес маршрутизатора (шлюза) устанавливается вручную при настройке сети), который, получив широковещательный ARP-запрос, сообщит компьютеру-адресату свой Ethernet-адрес и дальнейшая связь будет идти через маршрутизатор. Маршрутизатор анализирует свои таблицы маршрутизации, и на основании их принимает решение о маршруте доставки пакета. Таблицы маршрутизации частично составляются вручную администратором сети, а частично динамически обновляются, на основании данных соседних маршрутизаторов, по протоколам RIP, OSPF, NLSP, BGP и др. Таблица маршрутизации упрощенно выглядит следующим образом (в различных операционных системах и моделях маршрутизаторов возможны различные варианты):

Таблица

Пример таблицы маршрутизации.

	Адрес назначения (сеть или компьютер)	Маска подсети	Адрес следующего маршрутизатора (шлюза)	Метрика (расстояние до адресата)	Сетевой интерфейс
1	127.0.0.1	255.255.255.255	*	1	lo
2	210.1.1.0	255.255.255.0	*	1	eth0
3	130.30.0.0	255.255.0.0	*	1	eth1
4	190.55.0.0	255.255.0.0	*	1	eth2
5	170.10.0.0	255.255.0.0	130.30.10.5	1	eth1
6	13.1.10.17	255.255.255.255	130.30.10.5	1	eth1
7	200.15.1.0	255.255.255.0	130.30.10.7	1	eth1
8	200.15.1.0	255.255.255.0	190.55.15.1	3	eth2
9	0.0.0.0	0.0.0.0	231.1.1.5	1	ppp0

Данный пример составлен для компьютера (выполняющего функции шлюза и маршрутизатора), который подключен к сети 210.1.1.0 через сетевую карту eth0, имеет связь с Internet через модем (интерфейс ppp0), а также подключен к сети 130.30.0.0 1 через сетевую карту eth1, и к сети 190.55.0.0 через сетевую карту eth2 (см. рис).

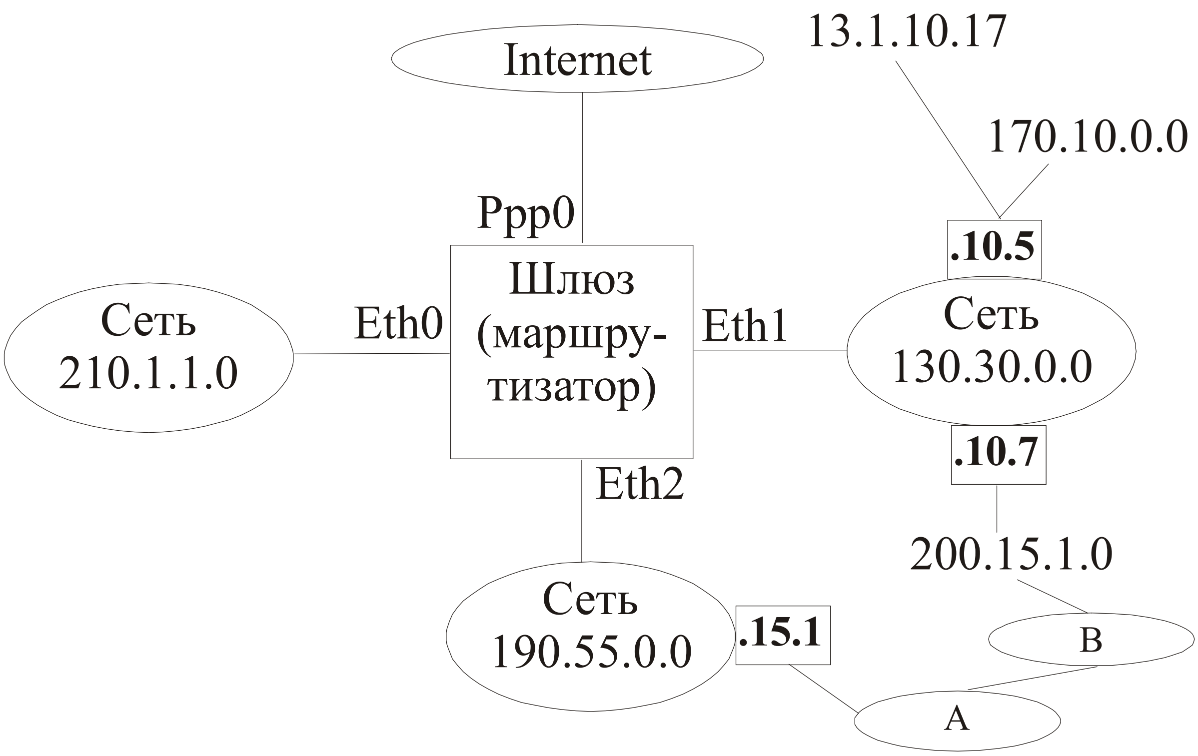


рис. Условная сеть для пояснения таблицы маршрутизации

Таблица

Пояснения к таблице маршрутизации

№ стр.	Пояснения
1	Описан loopback-адрес 127.0.0.1, т.е. ссылка на самого себя (фиктивный сетевой интерфейс lo).
2 - 4	Описываются сети, к которым непосредственно подключенные сетевые карты шлюза. Сеть 210.1.1.0 – к интерфейсу eth0, 130.30.0.0 – к интерфейсу eth1, 190.55.0.0 – к интерфейсу eth2. Доставка пакетов в эти сети происходит напрямую, поэтому в таблице адрес следующего маршрутизатора (шлюза) для них не указан.
5	Описан маршрут до сети 170.10.0.0. Все пакеты для компьютеров с адресами от 170.10.0.1 до 170.10.255.254 будут доставлены на маршрутизатор, описанный в таблице маршрутизатор с адресом 130.30.10.5 на интерфейсе Eth1.
6	Описан путь до единственного компьютера с адресом 13.1.10.17. Пакеты до него будут также доставляться на маршрутизатор 130.30.10.5 (интерфейс Eth1).

Продолжение табл.

№ стр.	Пояснения
7-8	Описаны два маршрута до сети 200.15.1.0, один из которых проходит через маршрутизатор 130.30.10.7 (интерфейс eth1), а второй – через маршрутизатор 190.55.15.1 (интерфейс eth2). Маршрут, проходящий через маршрутизатор 190.55.15.1 длиннее (метрика 3) и проходит через 3 сети: сеть 190.55.0.0, сеть A и сеть B. Указанный в таблице маршрутизации маршрутизатор 190.55.15.1 является лишь промежуточным: получив пакет до сети 200.15.1.0, он, в соответствии с собственной таблицей маршрутизации, передаст пакет маршрутизатору сети A. Тот проанализирует свою таблицу маршрутизации и передаст пакет маршрутизатору сети B, который на основании своей таблицы маршрутизации доставит пакет до сети назначения 200.15.1.0. Такая цепочечная схема доставки характерна для крупных сетей и позволяет не хранить на первом маршрутизаторе 190.55.15.1 информацию о всем пути следования пакета: достаточно только знать адрес ближайшего маршрутизатора на пути к адресату. Т.е. информация распределена между большим числом маршрутизаторов в сети. В противном случае, пришлось бы на каждом маршрутизаторе хранить все пути до всех существующих сетей, что нерационально, а в сети Internet и невозможно.
9	Описан маршрут по умолчанию 0.0.0.0. Любые пакеты до сетей (компьютеров) для которых не существует записи в таблице маршрутизации будут направлены по этому маршруту, т.е. в данном случае направлены в Internet, на маршрутизатор 231.1.1.5, соединенный с данным маршрутизатором по модему (сетевой интерфейс ppp0).

Необходимо также обратить внимание на поле "метрика" таблицы маршрутизации. Обычно метрика увеличивается на 1 при прохождении каждого маршрутизатора и соответствует реальному расстоянию до сети назначения, однако для особо перегруженных маршрутов маршрутизатор может быть вручную настроен так, чтобы увеличивать метрику более чем на 1, искусственно делая маршрут более длинным. В результате пакеты, если это возможно, будут направляться по другому, более короткому маршруту, и только пакеты для которых этот маршрут является единственным (или короче всех остальных) будут направлены на этот маршрутизатор.

4. Если в процессе доставки пакета возникнет ошибка, то будет получено сообщение по протоколу ICMP, указывающего причину ошибки. По протоколу ICMP может быть передана управляющая информация, позволяющая изменить маршрут доставки на более оптимальный или вообще поменять его, если какой-то шлюз временно не работает. Протокол ICMP также позволяет посылать короткие служебные пакеты (ping), которые позволяют протестировать работоспособность сети. Если с компьютера A будет послан ping компьютеру B, то операционная система компьютера B также ответит коротким пакетом по протоколу ICMP. После получения этого пакета компьютер A во-первых знает, что компьютер B доступен, а во вторых знает за какое время пакет дошел до компьютера B и вернулся обратно. После отправки нескольких ping-ов собирается статистика: минимальное, максимальное и среднее время приема-передачи пакетов, процент утерянных пакетов. Основные виды ICMP сообщений перечислены в таблице

Таблица

Основные виды ICMP сообщений.

Тип	Код	Сообщение
0	0	Echo Reply (Эхо-ответ)
3		Destination Unreachable (Адресат недостижим по различным причинам):
	0	Net Unreachable (нет маршрута в сеть)
	1	Host Unreachable (хост недоступен)
	2	Protocol Unreachable (протокол недоступен)
	3	Port Unreachable (порт недоступен)
	4	Datagram Too Big (необходима фрагментация, но она запрещена)
	5	Source Route Failed (невозможно выполнить опцию Source Route)
	13	Communication Administratively Prohibited (обработка дейтаграммы административно запрещена)
4	0	Source Quench (Замедление источника)
5		Redirect (выбрать другой маршрутизатор для посылки датаграмм):
	0	в данную сеть
	1	на данный хост
	2	в данную сеть с данным TOS (Type Of Service – тип обслуживания)
	3	на данный хост с данным TOS (Type Of Service – тип обслуживания)
8	0	Echo (Эхо-запрос)
9	0	Router Advertisement (Объявление маршрутизатора)

Продолжение табл.

Тип	Код	Сообщение
10	0	Router Solicitation (Запрос объявления маршрутизатора)
11		Time Exceeded (Время жизни дейтаграммы истекло)
	0	при передаче
	1	при сборке
12		Parameter problem (Ошибка в параметрах)
	0	Ошибка в IP-заголовке
	1	Отсутствует необходимая опция
13	0	Timestamp (Запрос временной метки для синхронизации часов)
14	0	Timestamp Reply (Ответ на запрос временной метки)
17	0	Address Mask Request (Запрос сетевой маски)
18	0	Address Mask Reply (Ответ на запрос сетевой маски)

2.3.2. Протоколы транспортного уровня TCP и UDP.

Протоколы транспортного уровня в стеке TCP/IP представлены двумя протоколами: TCP и UDP. Протокол TCP позволяет устанавливать виртуальный канал передачи данных между компьютерами. Канал устанавливается следующим образом:

1. Компьютер А посылает компьютеру В пакет, с установленным флагом SYN (синхронизация) и случайным числом (a)

   => SYN (a).

2. Компьютер В отвечает пакетом, с установленными флагами ACK (подтверждение), с параметром (a+1), и установленным флагом SYN и своим случайным числом (b).

   <= ACK(a+1), SYN (b)

3. Компьютер А завершает "рукопожатие" пакетом, с флагами ACK(a+1), ACK (b+1).

   => ACK (a+1), ACK (b+1)

После установления канала, программа может направлять в него данные непрерывным потоком, как на стандартное устройство ввода вывода. Протокол TCP сам разобьет данные на пакеты, при помощи алгоритма "скользящего окна" обеспечит подтверждение факта получения пакетов принимающей стороной и повторную передачу пакетов, если в этом будет необходимость. Кроме того, в протоколе TCP реализованы достаточно сложные механизмы регулирования загрузки сети и устранения заторов в сети. Протокол UDP более быстр, чем протокол TCP, однако менее надежен. Данные передаются без установления виртуального канала, в предположении, что принимающая сторона ждет данные. Программа должна сама позаботиться о разбитии передаваемых данных на пакеты, протокол не содержит средств подтверждения факта доставки сообщения и средств коррекции ошибок - все эти задачи должна решать программа.

При рассмотрении протоколов транспортного уровня необходимо остановиться на понятии "порт" и "сокет". Порт в протоколах транспортного уровня – это не физически существующий порт ввода-вывода (как, например, последовательный порт COM1), а "виртуальный" порт, который программно изолирует данные передаваемые по одному порту, от данных передаваемых по другому порту. Порты нумеруются от 0 до 65535. Существуют общеизвестные порты (well known ports), каждый из которых традиционно связан с тем или иным видом сетевого приложения. Например, стандартным портом для Web-сервера является порт 80. Большинство общеизвестных портов имеют номера меньше 1024. Это связано с тем, что в ОС Unix порты с номерами меньше 1024 доступны только приложениям с привилегиями суперпользователя root (администратор), поэтому пользователь без этих привилегий не сможет запустить собственный Web-сервер, который подменит на 80 порту настоящий Web-сервер. Порты TCP и порты UDP не зависят друг от друга. Порт 80 TCP может быть занят одним сетевым приложением, а 80 порт UDP – другим приложением.

Сокет (socket) – это описатель сетевого соединения между двумя сетевыми приложениями, которое включает в себя:

- IP-адрес и номер порта локальной машины.
- IP-адрес и номер порта удаленной машины.

Сокет однозначно описывает сетевое соединение. У двух различных соединений хотя бы один из приведенных выше параметров должен отличаться. Например к 80 порту сервера могут одновременно подключиться два приложения, работающие с различных портов на клиентской машине.

3. Протоколы прикладного уровня HTTP, FTP, SMTP, IMAP, POP3, TELNET.

В соответствии с архитектурой клиент-сервер, программа делится на две части (одна работает на сервере, вторая – на компьютере пользователя), функционирующие как единое целое. Протоколы прикладного уровня описывают взаимодействие клиентской и серверной частью программы. Выделяют следующие наиболее известные прикладные протоколы:

1. HTTP (Hyper Text Transfer Protocol)- протокол передачи гипертекста, работает на 80 порту. Используется в WWW для передачи гипертекстовых HTML страниц. При работе по этому протоколу, каждый элемент HTML – страницы загружается отдельно, причем соединение между загрузками прерывается и никакой информации о соединении не сохраняется. Это сделано для того, чтобы пользователя Web-страниц каждый получал "по чуть-чуть, в порядке общей очереди". В противном случае могла бы создаться ситуация, когда один человек качает страницу с большим количеством рисунков высокого разрешения, а все остальные ждут пока он это закончит.
2. FTP (File Transfer Protocol.) – протокол передачи файлов, работает на 20 и 21 порту. Предназначен для копирование файлов между компьютерами. Полностью занимает канал, пока не будет получен файл, сохраняет информацию о соединении. При сбое возможна докачка с того места, где произошел сбой.
3. SMTP, IMAP-4, POP3 – почтовые протоколы (электронная почта). SMTP - 25 порт, IMAP-4 – 143 порт, POP3 – 110 порт. Отличие: SMTP – протокол расчитанный на доставку почты до конкретного получателя, POP3 и IMAP-4 – протоколы взаимодействия пользователя со своим почтовым ящиком на сервере. При использовании SMTP предполагается, что почтовый адрес указывает на компьютер конечного получателя, и на этом компьютере запущена специальная программа, которая принимает и обрабатывает почту. Однако чаще всего бывает, что почта не доставляется на компьютер каждого отдельного пользователя, а обрабатывается централизованно, на отдельном почтовом сервере. В таком случае, каждый пользователь имеет на почтовом сервере свой почтовый ящик. Почта доставляется до сервера по протоколу SMTP (конечный получатель – сервер) и помещается в почтовые ящики пользователей. Затем пользователи подключаются к своим почтовым ящикам по протоколу POP3 или IMAP-4 и забирают почту. Протокол POP3 требует полностью скачать себе всю почту, а затем разбираться: нужна она вам была или нет. Причем, чаще всего, администратор запрещает хранить копии скачанной почты на сервере (или ограничивает время хранения копий), поэтому, например, скачав почту из почтового ящика на институтский компьютер, вы полностью очистите свой почтовый ящик и, зайдя на почтовый ящик с домашнего компьютера, увидите сообщение "Писем нет". Протокол IMAP-4 позволяет просматривать на сервере заголовки писем (указывается статус письма: новое, отвеченное и т.п.) и скачивать с сервера только необходимые письма или даже часть некоторого письма. Также можно на стороне сервера проводить поиск по сообщениям, создавать иерархию каталоговдля хранения полученных писем (копии скачанных писем остаются на сервере, пока вы их не удалите). Фактически IMAP4 дублирует функции почтовых программ пользователя (например, Microsoft Outlook), однако существенной разницей здесь являеется то, что если Micrsoft Outlook работает на компьютере пользователя, то команды протокола IMAP-4 выполняются на сервере, а значит каталоги с письмами хранятся в одном месте (на сервере), что очень удобно если вы часто подключаетесь к серверу с разных компьютеров и не хотите на каждом компьютере иметь полную копию всех писем.

   Резюмируя вышесказанное можно привести наиболее распространенный вариант работы с почтой для обычного пользователя: отправка почты – по протоколу SMTP (на почтовый сервер получателя), получение почты – по протоколу POP3 или IMAP-4 (скачивание почты из почтового ящика на своем почтовом сервере).
4. TELNET – используется для подключения и управления удаленным компьютером, работает на 23 порту. После подключения каждый символ, введенный на локальной машине, обрабатывается так, как если бы он был введен на удаленной машине. Либо может использоваться командный режим – управление удаленной машиной при помощи специальных команд. Фактически TELNET – это протокол эмуляции терминала: при помощи TELNET можно подключиться, например, на 25 порт и вручную набрать все необходимые поля заголовка письма, изменив адрес отправителя (обычно эти поля заполняются автоматически специальными почтовыми программами) и отправить само письмо. Или, например, подключиться на 80 порт и "поиграть" роль Web-браузера Internet Explorer.

4. Система доменных имен DNS.

Доменное имя – это имя компьютера, вида www.sait.com. Адресация в Internet происходит по IP-адресам, однако для человека гораздо удобнее доменные имена.

> *Существует также термин URL-адрес (Universal Resource Locator), т.е. запись вида http://www.sait.com, или в полном варианте http://www.sait.com:80/katalog/index.html#glava1. Доменное имя являеется частью URL-адреса (схема_передачи:// доменное_имя : порт / имя файла# внутреняя_ссылка).*

Встает проблема: как поставить в соответствие IP-адрес и доменное имя. Вести на каждом компьютере базу данных, содержащую все доменные имена Internet, невозможно, поэтому применяется служба доменных имен DNS (Domain Name Service). Алгоритм ее функционирования таков:
1. Пользователь в окне Web-браузера вводит http://www.microsoft.com
2. На первый DNS сервер IP-адрес которого известен (устанавливается в настройке Windows вручную или автоматически провайдером, при подключении к нему) компьютером пользователя направляется запрос на установление IP-адреса по доменному имени.

3. Если в базе данных сервера имеется соответствующая запись доменное имя – IP адрес, то ответ в виде IP-адреса возвращается компьютеру пользователю. Если в базе данных информация отсутствует, то запрос передается на DNS – сервер более высокого уровня (его IP известен серверу), который скорее всего тоже не знает ответ, но зато знает какой DNS сервер более низкого уровня отвечает за данную зону доменных имен, и перенаправит запрос ему. Тот ответит, и запрос по цепочке вернется к компьютеру пользователя. Такая схема наиболее распространена, однако возможна и другая схема. Если в базе данных сервера отсутствует запрашиваемая запись доменное имя – IP адрес, то компьютеру пользователя будет возвращен IP-адрес DNS-сервера более высокого уровня, и компьютер пользователя должен впоследствии сам выполнять запросы к последующим DNS-серверам.

Нет однозначного соответствия между IP–адресом и доменным именем. Компьютер, имеющий один и тот же IP-адрес, может иметь доменное имя www.minsk.by www.usa.com nowhere.ru и т.д. Для этого достаточно купить доменное имя, т.е. заплатить за регистрацию соответствующего IP-адреса в базе данных DNS – серверов, отвечающих за соответсвующие зоны имен. При этом сам компьютер может физически находиться хоть в Китае, или вообще, весь сайт может реально находиться на сервере, предоставляющем бесплатное размещение web-страниц (web-хостинг), в каталоге www.halyava.fi /pub/web/sait/5873, но вы купили доменное имя www.kruto.by и теперь пользователи могут попасть на ваш сайт, используя это имя.

За каждую зону имен отвечает минимум два DNS-сервера. Записи базы данных DNS-сервера хранятся в файле зоны, в формате, определяемом стандартом RFC-1035. Существует несколько типов записей для хранения раз личных данных. Рассмотрим эти записи подробнее.

Запись типа SOA.

SOA (Start of Authority) - начало зоны. Первая запись в базе данных зоны. Пример:

```
exmpl.ru.  IN  SOA   ns.exmpl.ru.   hostmaster.ns.exmpl.ru.  (
           1997120802
           10800
           3600
           3600000
           86400 )
```

Таблица

Расшифровка полей записи SOA

Поле	Значение
exmpl.ru.	Полностью уточненное имя зоны. Полностью уточненное доменное имя должно оканчиваться точкой. Если в файле зоны какое-либо другое имя не заканчивается на точку, то к данному имени добавляется имя зоны, т.е. имя server читается, как server.exmpl.ru.
SOA	Начало зоны
ns.exmpl.ru.	Доменное имя первичного DNS-сервера зоны.
hostmaster.ns.exmpl.ru.	Адрес электронной почты администратора DNS-сервера. Символ @ заменяется на точку, т.е. данный адрес выглядит, как hostmaster@ns.exmpl.ru.
1997120802	Серийный номер версии данных (выбор номера произволен, но номер должен увеличиваться после каждой модификации данных; обычно используется формат *YYYYMMDDVV,* где *YYYY* —год, *MM*— месяц, *DD* — день и *VV* — порядковый номер модификации зоны в указанный день).
10800	Период запросов на обновление данных со стороны вторичного сервера (в секундах).
3600	период повторов попыток запроса данных вторичным сервером в случае неудачи первого запроса (в секундах)
3600000	срок годности данных, то есть время, через которое вторичный сервер прекратит обслуживать запросы, если ему не удастся восстановить связь с первичным сервером (в секундах)
86400	время жизни данных зоны в кэше запросившего их сервера (в секундах)

Запись типа NS.

NS (Name Server) — указание серверов DNS, обслуживающих запросы к данной зоне имен. Указывается как первичный, так и вторичные сервера DNS. Пример:

```
exmpl.ru.        IN   NS      ns.exmpl.ru.
                 IN   NS      server.eldorado.com.
```

Таблица

Расшифровка полей записи NS

Поле	Значение
exmpl.ru.	Имя зоны, для которой указываются обслуживающие ее DNS-сервера.
NS	Сервер имен, т.е. DNS-сервер.
ns.exmpl.ru.	Доменное имя DNS-сервера зоны. Данный DNS-сервер являеется первичным сервером, что следует из записи SOA.
server.eldorado.com.	Доменное имя DNS-сервера зоны. Данный DNS-сервер являеется вторичным сервером. Для повышения надежности рекомендуется, чтобы сервера DNS находились в разных IP-сетях (не путать IP-сеть с зоной имен DNS).

Запись типа А.

A (Address) — указание IP-адреса для соответствующего доменного имени. Для одного и того же IP адреса может быть определено несколько доменных имен (например, на одном компьютере работает несколько Web-серверов, или один и тот же Web-сервер имеет несколько имен). Одно и то же доменное имя может иметь несколько записей типа А (это соответствует случаю, когда один узел имеет несколько IP-интерфейсов). Пример:

```
ns      IN  A   1.16.195.2
mail    IN  A   1.16.195.1
wildcat IN  A   1.16.195.3
```

Так как имена в примере не полностью уточненные, то к ним добавляется имя зоны **exmpl.ru,** то есть эти записи эквивалентны следующим:

```
ns.exmp1.ru.         IN  A   1.16.195.2
mail.exmpl.ru.       IN  A   1.16.195.1
wildcat.exmpl.ru.    IN  A   1.16.195.3
```

Запись типа CNAME.

CNAME (Canonical Name) — определение псевдонимов для доменных имен. В примере, приведенном ниже, доменное имя www.exmpl.ru. соответствует имени wildcat.exmpl.ru. Пример:

```
www   IN    CNAME     wildcat.exmpl.ru.
```

Запись типа MX.

MX (Mail Exchanger) — указание почтового сервера. Очень часто для обработки почты выделяется отдельный почтовый сервер, который получает, обрабатывает и хранит почту пользователей. Пример:

```
pochta.exmpl.ru.       IN  MX    10    mai1.exmp1.ru.
                       IN  MX    20    wildcat.exmpl.ru.
specpochta.exmpl.ru.   IN  MX    10    wildcat.exmpl.ru.
```

Таблица

Расшифровка полей записи MX

Поле	Значение
pochta.exmpl.ru.	Имя почтового домена, для которого указывается обслуживающий его почтовый сервер. Все почтовые сообщения, адресованные на адреса вида ящик@pochta.exmpl.ru. , будут отправляться для обработки на почтовые сервера, указанные ниже. Самого компьютера, с именем pochta.exmpl.ru., может и не существовать вообще.
MX	Почтовый сервер.
10	Приоритет почтового сервера. Одно и то же доменное имя может иметь несколько записей MX, указывающих на разные обработчики почты. Сначала будет предпринята попытка доставить почту на сервер с наименьшим числом в поле приоритета (в нашем случае, на mai1.exmp1.ru.). Если связь с данным сервером установить не удастся, то почта будет доставлена на компьютер wildcat.exmp1.ru., где она будет находится до тех пор, пока связь с компьютером mai1.exmp1.ru. не будет восстановлена. После восстановления связи, почта будет доставлена на mail.exmpl.ru. Компьютер wildcat.exmpl.ru. в данном случае выступает просто как почтовый ретранслятор, что необходимо для повышения надежности доставки почты.
mail.exmpl.ru.	Почтовый сервер для почтового домена pochta.exmpl.ru.
wildcat.exmpl.ru.	Почтовый ретранслятор (mail relay) для почтового домена pochta.exmpl.ru.
specpochta.exmpl.ru.	Имя почтового домена. Все почтовые сообщения, адресованные на адреса вида ящик@specpochta.exmpl.ru. будут направлены на обработку серверу wildcat.exmpl.ru.

При отсутствии записи MX для какого-либо доменного имени почта, адресованная на это доменное имя, будет доставляться непосредственно на хост, имеющий такое имя. Только каноническое доменное имя (не псевдоним) может быть указано в качестве обработчика почты.

Пример файла базы данных зоны exmpl.ru.

```
exmpl.ru.           IN     SOA   ns.exmpl.ru.    hostmaster.ns.exmpl.ru.  (
                           1997120802
                           10800
                           3600
                           3600000
                           86400 )
exmpl.ru.                  IN    NS             ns.exmpl.ru.
                           IN    NS             server.eldorado.com.
pochta.exmpl.ru.           IN    MX   10        mail.exmpl.ru.
                           IN    MX   20        wildcat.exmpl.ru.
specpochta.exmpl.ru.       IN    MX   10        wildcat.exmpl.ru.
mail                       IN    A              1.16.195.1
wildcat                    IN    A              1.16.195.3
ns                         IN    A              1.16.195.2
www                        IN    CNAME          wildcat.exmpl.ru.
```

Файл базы данных обратной зоны

Если файл базы данных прямой зоны служит для преобразования доменных имен в IP-адреса, то файл базы данных обратной зоны используется для преобразования IP-адресов в доменные имена. Название обратной зоны строится по следующему принципу: в обратном порядке записывается сетевая часть IP-адреса, к которой добавляется стандартный домен in-addr.arpa. Например, для сети 1.16.195.0 обратная зона будет иметь название 195.16.l.in-addr.arpa. В файле базы данных обратной зоны используются следующие типы записей.

Запись типа PTR

PTR (Pointer) — указатель на доменное имя. В примере, приведенном ниже, указывается, что IP-адрес 1.16.195.1 соответствует доменному имени ns.exmpl.ru. Пример (для зоны 195.16.l.in-addr.arpa):

```
98      IN      PTR      ns.exmpl.ru.
```

Или, полностью уточняя *имя* в левой части:

Запись типа A

Запись типа A имеет специальное значение для имен из домена in-addr.arpa и указывает маску подсети. Пример (в базе данных зоны 195.16.1 .in-addr.arpa):

```
0       IN      A       255.255.255.0
```

Пример базы данных обратной зоны

```
195.16.1.in-addr.arpa.   IN     SOA    ns.exmpl.ru.     hostmaster.ns.exmpl.ru. (
                1997120802
                10800
                3600
                3600000
                86400 )
        IN      NS      ns.exmpl.ru.
        IN      NS      server.eldorado.com.
0       IN      A       255.255.255.0
1       IN      PTR     mail.exmpl.ru.
2       IN      PTR     ns.exmpl.ru.
3       IN      PTR     wildcat.exmpl.ru.
```

Лекция 3. Сетевое оборудование.

3.1. Повторитель (концентратор, hub)

В начале 80-х годов сети Ethernet организовывались на базе шинной топологии с использованием сегментов на основе коаксиального кабеля. С увеличением длины кабеля, соединяющего компьютеры, усиливается затухание сигнала в кабеле, поэтому максимальная длина кабеля соединяющего компьютеры в сети не может превышать 500 метров для толстого жесткого коаксиального кабеля и 185 метров для тонкого кабеля Ethernet. Таким образом, максимально возможная общая длина всех кабелей сети – 500 метров. Для преодоления 500-метрового барьера используют повторители (repeater). Повторитель просто по битам копирует (пересылает) все пакеты Ethernet из одного сегмента сети во все другие, подключенные к нему (см. рис.1).



Рис. 1. Двухпортовый повторитель (схема).

Повторитель работает на физическом уровне модели OSI. Основной задачей повторителя является восстановление электрических сигналов для передачи их в другие сегменты. За счет усиления и восстановления формы электрических сигналов повторителем, становится возможным расширение сетей, построенных на основе коаксиального кабеля и увеличение общего числа пользователей сети.

Повторители бывают 2-х и многопортовыми. Двухпортовые повторители (см. рисунок 1) используются в сетях с шинной топологией, построенных на коаксиальном кабеле. Многопортовые повторители используются в сетях с топологией типа "звезда" (кабель "витая пара"). И 2-х и многопортовые повторители, получив пакет на один из своих портов, просто передает его во все остальные порты.

Многопортовые повторители, в сетях построенных на кабеле "витая пара", часто называют концентраторами или хабами (Hub). Хабы нужны даже не столько для усиления сигнала, как для соединения в сеть более чем двух компьютеров, т.к. кабель "витая пара" позволяет напрямую соединить только два компьютера. Если же необходимо соединить три компьютера, то каждый из них напрямую подключается к хабу, который и ретранслирует сигнал, полученный от одного компьютера на все остальные порты, к которым подключены другие компьютеры (см. рис. 2 и 3).

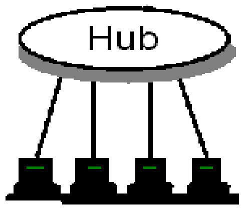


Рис. 2 Многопортовый повторитель (Hub) – схема подключения компьютеров по топологии "звезда".

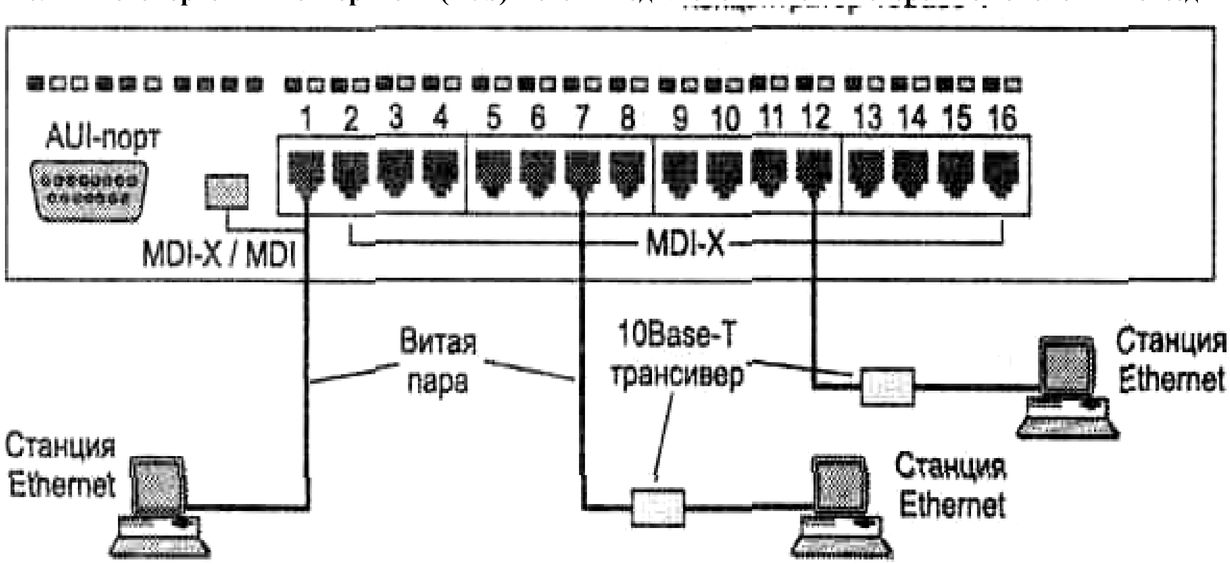


Рис. 3 Концентратор (hub) – внешний вид (схема).

Различия в моделях концентраторов при выполнении основной функции (побитное дублирование сигнала на все порты) невелики и, в основном, зависит от типа кабеля (витая пара, оптоволоконный и т.п.). Однако различные модели концентраторы могут реализовывать и дополнительные функции, некоторые из которых рассмотрены ниже.

Отключение портов

Концентраторы способны отключать некорректно работающие порты, изолируя тем самым остальную часть сети от возникших в узле проблем. Эту функцию называют *автосегментацией (autopartitioning).* Отключение происходит при отсутствии ответа на последовательность импульсов link test (проверка связи), посылаемых во все порты каждые 16 мс. В этом случае неисправный порт переводится в состояние "отключен", но импульсы link test будут продолжать посылаться в порт с тем, чтобы при восстановлении устройства работа с ним была продолжена автоматически. Отключение порта может произойти и по другим причинам:

- *Ошибки на уровне кадра.* Если интенсивность прохождения через порт кадров, имеющих ошибки, превышает заданный порог, то порт отключается, а затем, при отсутствии ошибок в течение заданного времени, включается снова. Такими ошибками могут быть: неверная контрольная сумм, неверная длина кадра (больше 1518 байт или меньше 64 байт), неоформленный заголовок кадра.
- *Множественные коллизии.* Если концентратор фиксирует, что источником коллизии был один и тот же порт 60 раз подряд, то порт отключается. Через некоторое время порт снова будет включен.
- *Затянувшаяся передача (jabber).* Если время прохождения одного кадра через порт превышает время передачи кадра максимальной длины в 3 раза, то порт отключается.

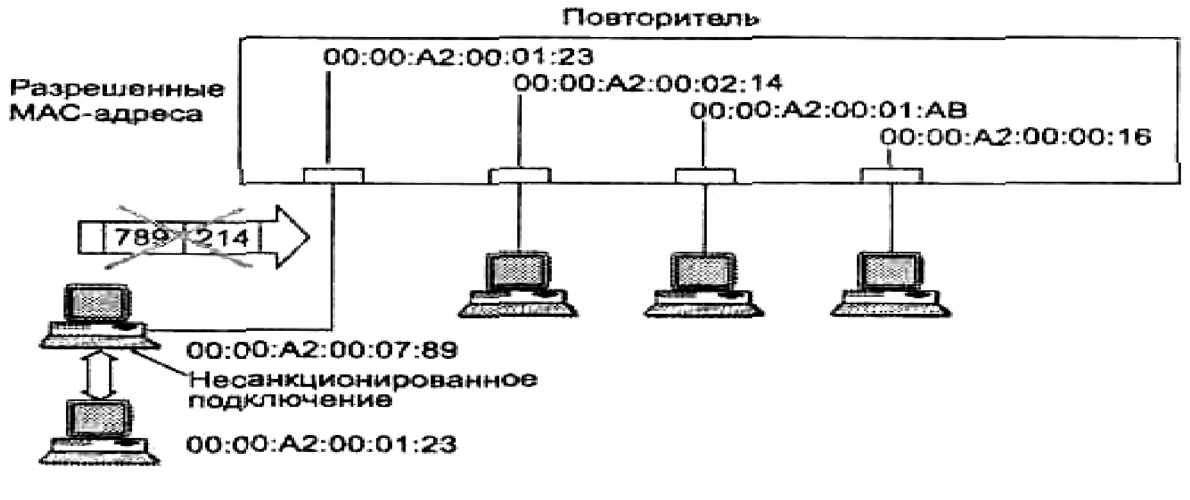
Поддержка резервных связей

Так как использование резервных связей в концентраторах определено только в стандарте FDDI, то для остальных стандартов разработчики концентраторов поддерживают эту функцию с помощью своих частных решений. Например, в концентраторах Ethernet/Fast Ethernet резервные связи всегда должны соединять отключенные порты, чтобы не нарушать логику работы сети. При конфигурировании концентратора администратор должен определить, какие порты являются основными, а какие по отношению к ним — резервными. Если по какой-либо причине порт отключается (срабатывает механизм автосегментации), концентратор делает активным его резервный порт. В некоторых моделях концентраторов разрешается использовать механизм назначения резервных портов только для оптоволоконных портов, считая, что нужно резервировать только наиболее важные связи, которые обычно выполняются на оптическом кабеле.

Защита от несанкционированного доступа

Общая разделяемая среда предоставляет очень удобную возможность для несанкционированного прослушивания сети и получения доступа к передаваемым данным. Для этого достаточно подключить компьютер с программным анализатором протоколов (сниффером - sniffer) к свободному разъему концентратора, записать на диск все проходящие по сети кадры, а затем выделить из них нужную информацию.

Разработчики концентраторов предоставляют различные способы защиты данных в разделяемых средах. Наиболее простой способ защиты заключается в том, что администратор вручную связывает с каждым портом концентратора некоторый MAC-адрес. Этот MAC-адрес является адресом сетевой карты компьютера, которому разрешено подключаться к данному порту. Например, на рис. первому порту концентратора назначен MAC-адрес 01:23 (условная запись). Компьютер с MAC-адресом сетевой карты 01:23 нормально работает с сетью через данный порт. Если злоумышленник отсоединяет этот компьютер и присоединяет вместо него свой, концентратор заметит, что при старте нового компьютера в сеть начали поступать кадры с адресом источника 07:89. Так как этот адрес является недопустимым для первого порта,



Другим способом защиты данных является случайное искажение данных в кадрах, передаваемых портам с адресом, отличным от адреса назначения пакета. При этом методе каждому порту концентратора также ставится в соответствие некоторый MAC-адрес сетевой карты. Кадр, поступивший на концентратор, дублируется на все порты, как этого и требует стандарт. При этом заголовки сдублированных кадров остаются неизменными, а в поле данных кадров помещаются нули. Полезные данные сохраняются только в поле данных кадра, направленного на порт, к которому подключен компьютер-адресат. Этот метод сохраняет логику случайного доступа к среде, так как все компьютеры видят, что сеть занята кадром, предназначенным одному из них (заголовок кадра не заполняется нулями), но только станция, которой послан этот кадр, может понять содержание поля данных кадра (см. рис.).

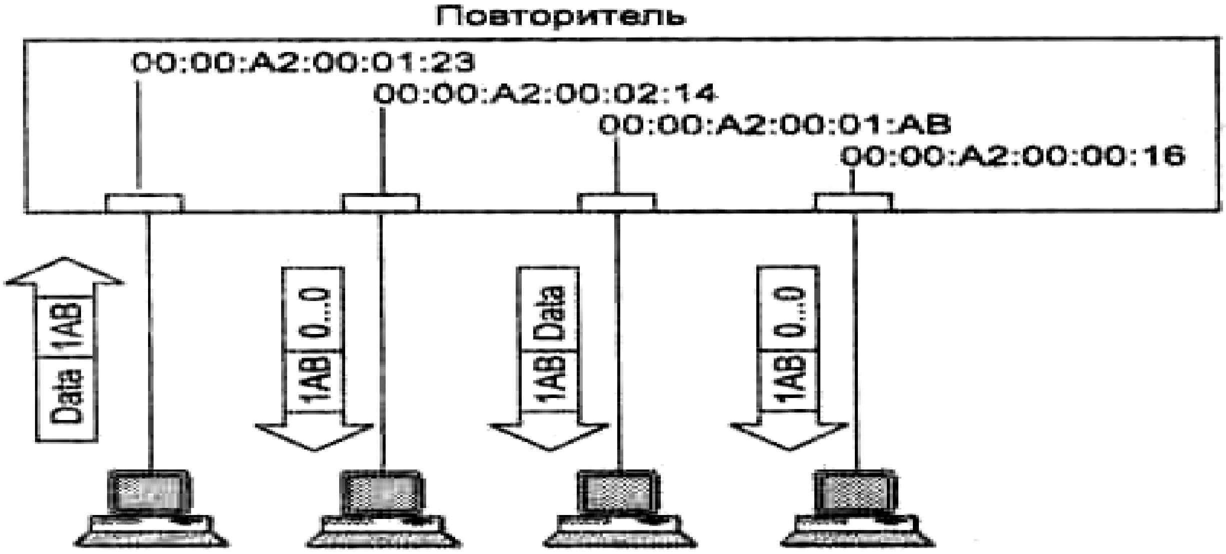


Рис. Искажение поля данных в кадрах, не предназначенных для приема станциями

Для реализации описанных выше методов защиты концентратор нужно предварительно сконфигурировать. Для этого концентратор должен иметь блок управления. Концентраторы, имеющие блок управления, обычно называют интеллектуальными (smart-hub). Блок управления представляет собой компактный вычислительный блок со встроенным программным обеспечением. Для взаимодействия администратора с блоком управления концентратор имеет консольный порт (чаще всего RS-232), к которому подключается терминал или персональный компьютер с программой эмуляции терминала. При присоединении терминала блок управления выдает на экран некоторое меню, с помощью которого администратор выбирает нужное действие и конфигурирует концентратор.

Многосегментные концентраторы

Многосегментные концентраторы обычно имеют большое количество портов (например, 72 или 240). Очевидно, что разделять среду передачи данных между таким количеством компьютеров нерационально. Поэтому в таких концентраторах имеется несколько несвязанных внутренних шин передачи данных, которые предназначены для создания нескольких разделяемых сред. Например, концентратор, изображенный на рис. , имеет три внутренние шины Ethernet. Первые два компьютера связаны с шиной Ethernet 3, а третий и четвертый компьютеры — с шиной Ethernet 1. Первые два компьютера образуют один разделяемый сегмент, а третий и четвертый — другой разделяемый сегмент.

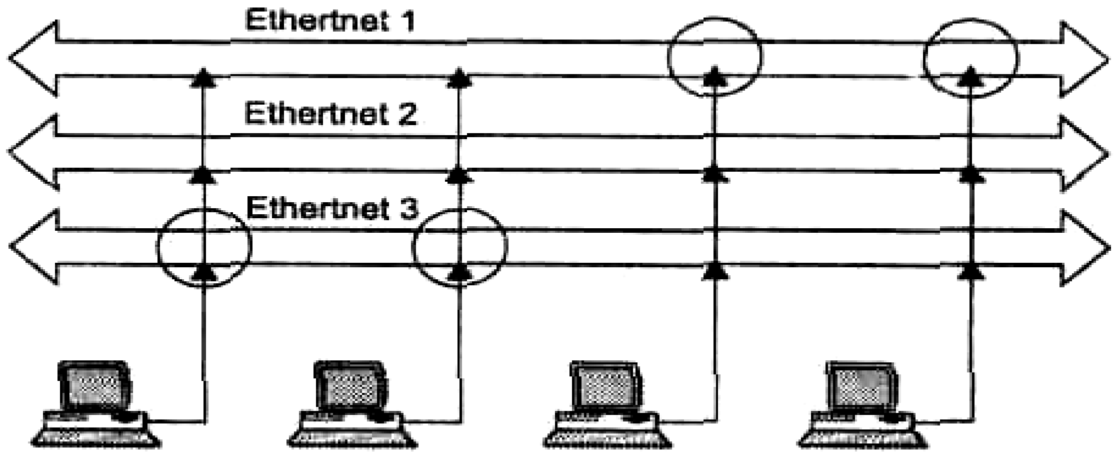


Рис. Многосегментный концентратор.

Между собой компьютеры, подключенные к разным сегментам, общаться через концентратор не могут, так как шины внутри концентратора никак не связаны. Для объединения сегментов необходимо использовать дополнительные сетевые устройства (мосты, коммутаторы, маршрутизаторы – см. дальше в лекциях). Многосегментные концентраторы нужны для создания разделяемых сегментов, состав которых может легко изменяться. Большинство многосегментных концентраторов, например System 5000 компании Nortel Networks или PortSwitch Hub компании 3Com, позволяют выполнять операцию соединения порта с одной из внутренних шин чисто программным способом, например с помощью локального конфигурирования через консольный порт. В результате администратор сети может присоединять компьютеры пользователей к любым портам концентратора, а затем с помощью программы конфигурирования концентратора управлять составом каждого сегмента. Если завтра сегмент Ethernet1 станет перегруженным, то его компьютеры можно распределить между оставшимися сегментами концентратора. Возможность многосегментного концентратора программно изменять связи портов с внутренними шинами называется конфигурационной коммутацией (configuration switching).

Управление концентратором по протоколу SNMP

Как видно из описания дополнительных функций, многие из них требуют конфигурирования концентратора. Это конфигурирование может производиться локально, путем подключения персонального компьютера или терминала к концентратору через интерфейс RS-232C, однако при большом количестве концентраторов в сети это становится неудобным. Поэтому большинство концентраторов, поддерживающих интеллектуальные дополнительные функции, могут управляться централизованно по сети с помощью протокола управления сетью SNMP (Simple Network Management Protocol) из стека TCP/IP.

В блок управления концентратором встраивается так называемый SNMP-агент, который имеет свой MAC- и IP-адрес. SNMP-агент собирает информацию о состоянии концентратора и хранит ее в базе данных управляющей информации — Management Information Base (MIB) – блока управления, которая позволяет одному из компьютеров сети, выполняющему роль центральной станции управления, запрашивать у SNMP-агента значения стандартных переменных базы MIB. В переменных хранятся не только данные о состоянии концентратора, но и управляющая информация, воздействующая на него. Например, в MIB есть переменная, управляющая состоянием порта ("включить" – "выключить").

Конструктивное исполнение концентраторов

По конструктивным особенностям выделяют следующие типы концентраторов:

- концентраторы с фиксированным количеством портов
- модульные концентраторы
- стековые концентраторы
- модульно-стековые концентраторы

Концентратор с фиксированным количеством портов — это наиболее простое конструктивное исполнение, когда устройство представляет собой отдельный корпус со всеми необходимыми элементами (портами, органами индикации и управления, блоком питания), и эти элементы заменять нельзя. Обычно общее количество портов изменяется от 4-8 до 24. Один порт может быть специально выделен для подключения концентратора к другому концентратору или иметь кнопочный переключатель, позволяющий подключать к этому порту как обычный компьютер (маркировка MDI-X, см. рис. 3), так и другой концентратор (маркировка MDI). Концентратор также может иметь разъем AUI для соединения (при помощи трансивера) с толстым коаксиальным кабелем, концентратором оптоволоконных сетей или другим концентратором "витая пара".

Модульный концентратор выполняется в виде отдельных модулей с фиксированным количеством портов, устанавливаемых на общее шасси. Шасси имеет внутреннюю шину для объединения отдельных модулей в единый повторитель. Часто такие концентраторы являются многосегментными, тогда в пределах одного модульного концентратора работает несколько несвязанных между собой повторителей. Агент протокола SNMP обычно выполняется в виде отдельного модуля, при установке которого концентратор превращается в интеллектуальное устройство. Модульные снабжаются системой терморегулирования, избыточными источниками питания, позволяют осуществлять замену модулей без отключения питания и дают возможность быстро и гибко реагировать на изменения конфигурации сети. Недостатком модульных концентраторов на основе шасси является высокая начальная стоимость такого устройства, т.к. даже если установлено всего 1-2 модуля, концентратор поставляется вместе со всеми общими устройствами (избыточные источники питания и т. п). Поэтому для сетей средних размеров большую популярность завоевали стековые концентраторы.

Стековый концентратор, как и концентратор с фиксированным числом портов, выполнен в виде отдельного корпуса с фиксированным количеством портов. Однако стековыми эти концентраторы называются не потому, что они устанавливаются один на другой, в общую стойку. Стековые концентраторы имеют специальные порты и кабели для объединения нескольких корпусов в единый повторитель, который имеет общий блок повторения и, с точки зрения правила 4-х хабов, считается одним повторителем. Число объединяемых в стек корпусов может быть достаточно большим (обычно до 8, но бывает и больше). Выгодной чертой стековых концентраторов является их более низкая стоимость, так как сначала предприятие может купить

одно устройство без избыточного шасси, а потом нарастить стек еще несколькими аналогичными устройствами.

Модулъно-стековые концентраторы представляют собой модульные концентраторы, объединенные специальными кабелями в стек. Как правило, корпуса таких концентраторов рассчитаны на небольшое количество модулей (1-3). Эти концентраторы сочетают достоинства концентраторов обоих типов.

3.2. Мост (bridge)

Повторители, за счет усиления и восстановления формы электрических сигналов, позволяют увеличить протяженность сети, однако и здесь есть ограничения: из-за задержки приема-передачи сигнала в повторителе, между любыми двумя компьютерами в сети Ethernet не может быть более 4-х повторителей, а в сети Fast Ethernet – не более одного повторителя 1-го класса и не более двух повторителей 2-го класса (подробнее см. далее в лекциях). Поэтому для создания более протяженных сетей необходимо пользоваться дополнительными сетевыми устройствами – мостами (bridge).

Мосты позволяют преодолеть ограничение "не более четырех повторителей между любыми двумя компьютерами" за счет того, что работают не на физическом, а на канальном уровне модели OSI. Т.е. мост ретранслирует кадр не по битам, а полностью принимает кадр в свой буфер, ***заново*** получает доступ к разделяемой среде и ретранслирует кадр в сеть. Помимо увеличения протяженности сети, мост также позволяет разбить ее на сегменты с независимыми разделяемыми средами, увеличив общую пропускную способность сети. Поясним на примере: пусть имеется три повторителя (хаба), к каждому из которых, при помощи кабеля "витая пара", подключено по четыре компьютера (см. рис.). Повторители соединены между собой при помощи моста. Допустим компьютер К1 передает в сеть кадр сообщения для компьютера К4. Кадр сообщения по кабелю попадет на повторитель1, который сдублирует его на все свои порты, т.е. кадр сообщения получат компьютеры К2, К3, К4 (что и требовалось) и мост. Мост, получив кадр сообщения от повторителя, анализирует "адрес получателя", имеющийся в кадре, определяет что компьютер К4 относится к сегменту 1 и поэтому кадр сообщения не надо дублировать для повторителей 2 и 3 (если бы кадр сообщения относился к компьютеру К5, то мост передал бы этот кадр только повторителю 2, ничего не передавая на порт, к которому подключен повторитель 3).

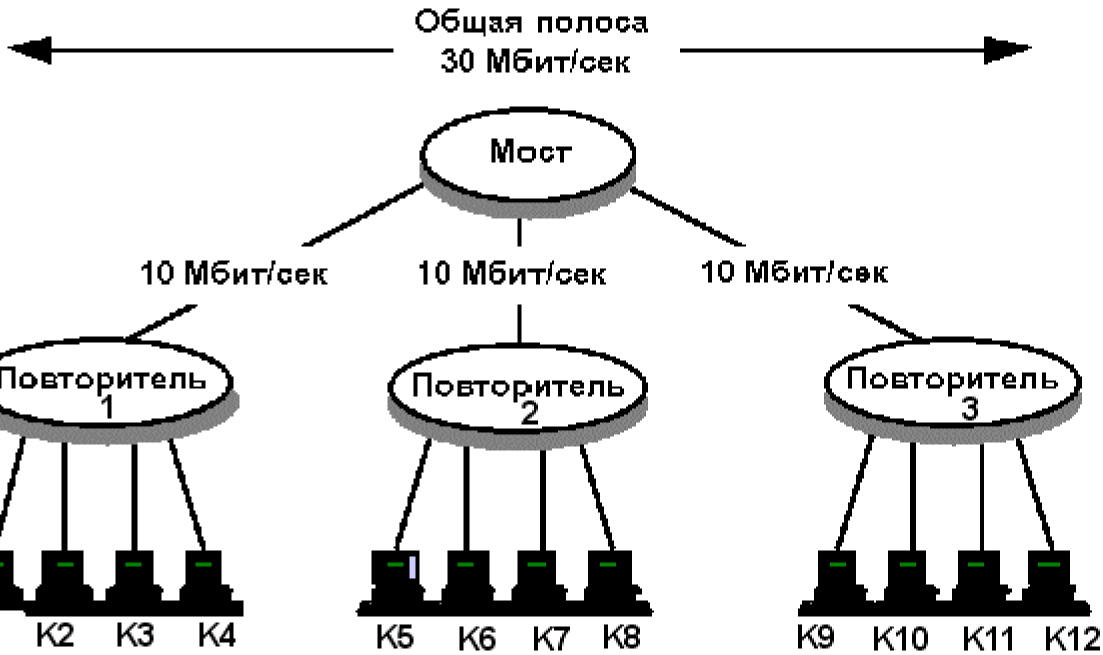


рис. Пояснение алгоритма работы моста.

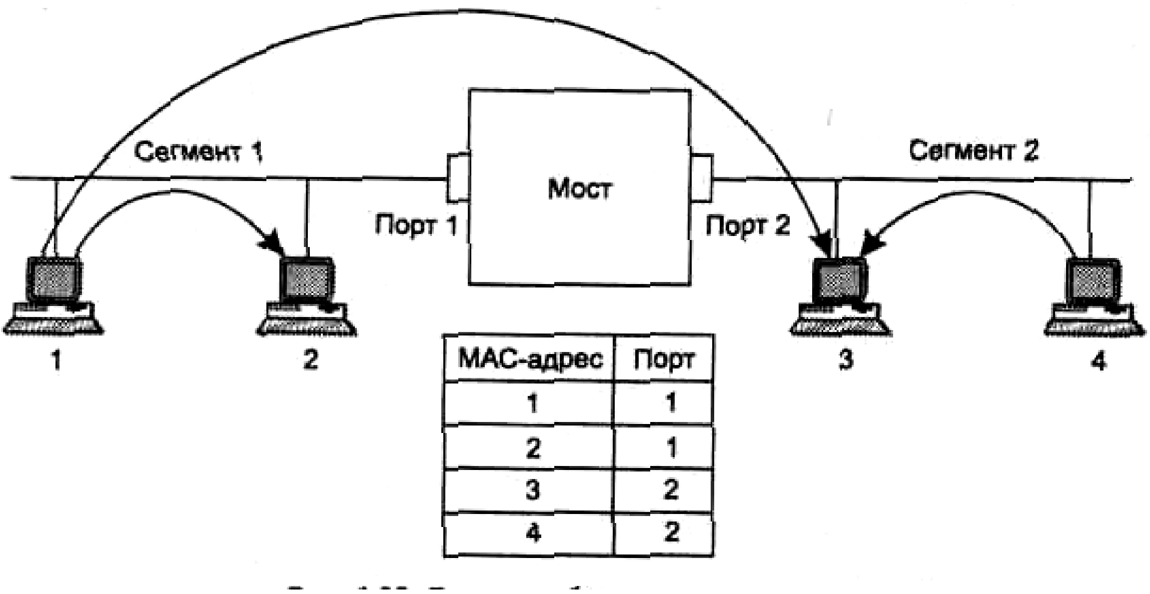
Рассмотрим какие выгоды дает такая схема. В момент когда компьютер К1 передает кадр сообщения, ни один из компьютеров К2-К4 не может ничего передавать в сеть – сеть "занята". Однако в тот же момент времени компьютеры К5-К12 могут передавать сообщения друг другу – для них сеть "свободна", т.к. мост не передал кадр сообщения от компьютера К1 в их сегменты сети. Таким образом, если файл копируется с компьютера К1 на К4 со скоростью 10 Мбит/с, с компьютера К5 на К8 со скоростью 10 Мбит/с, с компьютера К9 на К11 со скоростью 10 Мбит/с, то суммарная пропускная способность сети составляет 30 Мбит/с. Если бы вместо моста в вершине этой схемы стоял простой повторитель, то кадр сообщения от компьютера К1 "занял" бы всю сеть, и ни один из компьютеров К2-К12 не смог бы в это время передавать в сеть что-либо (без возникновения коллизии), а пропускная способность сети упала бы до 10 Мбит/с.

Существует два основных алгоритма работы моста: алгоритм прозрачного моста и алгоритм моста с маршрутизацией от источника. Алгоритм прозрачного моста используется в сетях Ethernet, а алгоритм моста

с маршрутизацией от источника может использоваться в сетях Token Ring и FDDI, хотя в этих сетях могут использоваться и обычные прозрачные мосты.

Алгоритм работы прозрачного моста.

Мост при таком алгоритме не заметен (прозрачен) для сетевых карт. Сетевая карта посылает кадр данных сетевой карте другого компьютера так, как если бы моста в сети и не было. Порты моста подключены к соединяемым сегментам сети и не имеют своих MAC-адресов, захватывая все проходящие по сети пакеты. Первоначально мост не знает к какому порту подключены какие компьютеры (см. рис.). Поэтому, если компьютер 1 направит кадр компьютеру 2, то мост, захватив этот пакет на порту 1, сдублирует его на все остальные порты, т.е. в данном случае на порт 2 (хотя по логике работы моста этого делать и не надо, но мост пока не знает к какому сегменту относится компьютер 2). Одновременно с этим, мост сделает в своей внутренней таблице адресов запись, что компьютер 1 относится к сегменту 1 (т.к. кадр от него был захвачен с порта 1), и кадры для компьютера 1 надо дублировать только на порт 1. Если все четыре компьютера достаточно активны в сети, то таблица адресов моста заполнится, и он будет дублировать кадры только на те порты, на которые это действительно необходимо. Таким образом, трафик между компьютерами 1 и 2 будет отделен от трафика между компьютерами 3 и 4, т.е. кадры от компьютера 1 к компьютеру 2 не будут дублироваться на порт 2.



MAC-адрес	Порт
1	1
2	1
3	2
4	2

рис. Алгоритм работы прозрачного моста

Каждая автоматически созданная запись о принадлежности компьютера к сегменту 1 или 2 имеет срок жизни. Если до истечения срока жизни запись не подтверждалась кадрами, проходящими по сети, то она помечается как недействительная. Если, в любой момент времени, компьютер 1 будет перемещен в сегмент 2 и пакеты от него станут поступать на порт 2, то соответствующая запись в таблице будет изменена. Помимо динамически создаваемых записей, могут существовать и статические записи, созданные администратором вручную, при конфигурировании моста, и не имеющие срока жизни. При помощи статических записей можно жестко описать принадлежность компьютера к тому или иному сегменту, или указать, что пакеты к компьютеру 1 должны всегда дублироваться на все порты (flood - затопление), а пакеты к компьютеру 2 никогда не должны дублироваться ни на какие порты (discard - отбросить).

Алгоритм работы моста с маршрутизацией от источника (SR-мосты).

Этот алгоритм используется в сетях Token Ring и FDDI. Компьютер-отправитель помещает в кадр всю адресную информацию о промежуточных мостах и кольцах, которые кадр должен пройти на пути к компьютеру-адресату. Первоначально компьютер-отправитель не имеет никакой информации о пути к компьютеру адресату. Кадр просто передается в кольцо, в надежде, что адресат находится в одном кольце с отправителем. Если компьютер-адресат в кольце отсутствует не так, то кадр сделает оборот по кольцу и вернется без установленного признака "кадр получен" (бит "адрес распознан" и бит "кадр скопирован"). В таком случае компьютер-отправитель пошлет одномаршрутный широковещательный кадр-исследователь (SRBF, Single Route Brodcast Frame). Этот кадр распространяется по сети: мосты дублируют кадр на все свои порты, за исключением заблокированных администратором (для избежания петлевых маршрутов и зацикливания кадра). В конце-концов кадр-иследователь будет получен компьютером-адресатом, который немедленно отправит многомаршрутный широковещательный кадр-исследователь (ARBF, All Route Brodcast Frame). Этот кадр распространяется по сети, дублируясь мостами на все порты без исключения (для предотвращения зацикливания, кадр-ARBF уже однажды сдублированный мостом на один из своих портов, заново

на этот порт не дублируется). В конце-концов, до компьютера-отправителя дойдет множество кадров-ARBF, прошедших через все возможные маршруты от компьютера-адресата до компьютера-исследователя. Полученная информация попадет компьютеру-отправителю и в маршрутные таблицы моста, соединяющего кольцо компьютера-отправителя с остальной сетью. Впоследствии все компьютеры этого кольца могут воспользоваться информацией моста при отправке своих кадров.

Ограничения топологии сетей, построенных на прозрачных мостах.

Основным ограничением при использовании мостов является отсутствие петлевых маршрутов. Поясним на

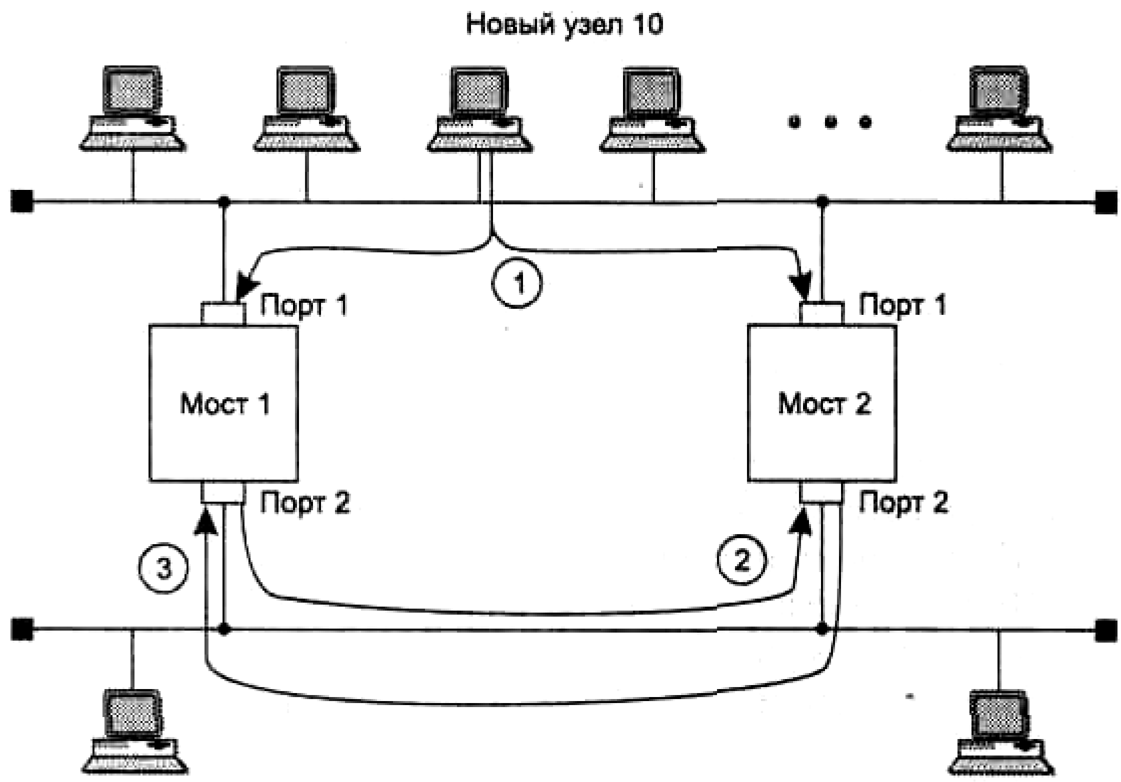


рис. Ошибки в работе мостов, возникающие при наличии петлеобразных маршрутов.

Пусть новый компьютер с адресом 10 впервые начинает работу в данной сети. Обычно начало работы любой операционной системы сопровождается рассылкой широковещательных кадров, в которых компьютер заявляет о своем существовании и одновременно ищет серверы сети. На этапе 1 компьютер посылает первый кадр с широковещательным адресом назначения и адресом источника 10 в свой сегмент. Кадр попадает как в мост 1, так и в мост 2. В обоих мостах новый адрес источника 10 заносится в адресную таблицу с пометкой о его принадлежности сегменту 1, то есть создается новая запись вида: MAC-адрес 10 – порт 1. Так как кадр, рассылаемый компьютером, имеет широковещательный адрес назначения, то каждый мост должен передать кадр на сегмент 2. Эта передача происходит поочередно, в соответствии с методом случайного доступа CSMA/CD технологии Ethernet. Пусть первым доступ к сегменту 2 получил мост 1 (этап 2). При ретрансляции мостом 1 кадра в сегмент 2 мост 2 принимает его в свой буфер и обрабатывает. Он видит, что адрес 10 уже есть в его адресной таблице, но пришедший кадр является более свежим, и он утверждает, что адрес 10 принадлежит сегменту 2, а не 1. Поэтому мост 2 корректирует содержимое базы и делает запись о том, что адрес 10 принадлежит сегменту 2. Аналогично поступает мост 1, когда мост 2 получит доступ к разделяемой среде и передает свою копию широковещательного кадра на сегмент 2. Результатом описанной ситуации является следующее:

- Размножение кадра, то есть появление нескольких его копий (в данном случае — двух, но если бы сегменты были соединены тремя мостами — то трех и т. д.).
- Бесконечная циркуляция обеих копий кадра по петле в противоположных направлениях, а значит, засорение сети ненужным трафиком.
- Постоянная перестройка мостами своих адресных таблиц, так как кадр с адресом источника 10 будет появляться то на одном порту, то на другом.

Чтобы исключить все эти нежелательные эффекты, мосты нужно применять так, чтобы между логическими сегментами не было петель, то есть строить с помощью мостов только древовидные структуры, гарантирующие наличие только одного пути между любыми двумя сегментами. В простых сетях сравнительно легко гарантировать существование одного и только одного пути между двумя сегментами. Но когда количество соединений возрастает и сеть становится сложной, то вероятность непреднамеренного образования петли

оказывается высокой. Кроме того, желательно для повышения надежности иметь между мостами резервные связи, которые не участвуют при нормальной работе основных связей в передаче информационных пакетов станций, но при отказе какой-либо основной связи образуют новую связную рабочую конфигурацию без петель. Поэтому в сложных сетях между логическими сегментами прокладывают избыточные связи, которые образуют петли, но для исключения активных петель блокируют некоторые порты мостов. Наиболее просто эта задача решается вручную, но существуют и алгоритмы, которые позволяют решать ее автоматически. Наиболее известным является стандартный алгоритм покрывающего дерева (Spanning Tree Algorithm, STA). Кроме того, имеются фирменные алгоритмы, решающие ту же задачу, но с некоторыми улучшениями для конкретных моделей мостов и коммутаторов.

Удаленные мосты

Удаленный мост – это мост, который через один или несколько портов подключен к глобальной сети (Internet, X.25, FrameRelay, ATM). Удаленные мосты (а также удаленные маршрутизаторы) используются для соединения локальных сетей через глобальные сети. Если в локальных сетях мосты постепенно вытесняются коммутаторами, то удаленные мосты и сегодня продолжают пользоваться популярностью. Удаленные мосты не надо конфигурировать (адресная таблица строится автоматически), а при объединении с сетью предприятия сетей филиалов, где нет квалифицированного обслуживающего персонала, это свойство оказывается очень полезным.

Как и в локальных сетях, важной характеристикой удаленных мостов (удаленных маршрутизаторов) является скорость обработки кадров, которые часто ограничиваются не внутренними возможностями устройства, а скоростью передачи данных по линии (например, аналоговой телефонной линии). Для преодоления ограничений на скорость линии, а также для уменьшения части локального трафика, передаваемого по глобальной линии, в удаленных мостах и маршрутизаторах используются специальные приемы, отсутствующие в локальных устройствах. Эти приемы не входят в стандарты протоколов, но они реализованы практически во всех устройствах, обслуживающих низкоскоростные каналы, особенно каналы со скоростями в диапазоне от 9600 бит/с до 64 Кбит/с. К таким приемам относятся технологии сжатия пакетов, спуфинга и сегментации пакетов.

Сжатие пакетов (компрессия). Некоторые производители, используя собственные алгоритмы, обеспечивают коэффициент сжатия до 8:1. Стандартные алгоритмы сжатия, применяемые в модемах, обеспечивают коэффициент сжатия до 4:1. После сжатия данных для передачи требуется существенно меньшая скорость канала.

Спуфинг (spoofing). Эта технология позволяет значительно повысить пропускную способность линий, объединяющих локальные сети, работающие по протоколам с большим количеством широковещательных пакетов. Широковещательные пакеты характерны для большинства сетевых операционных систем, за исключением ОС Unix, которая изначально строилась для медленных глобальных линий связи. Главной идеей спуфинга является имитация передачи пакета по глобальной сети. Рассмотрим технику спуфинга на примере передачи между удаленными сетями пакетов SAP (Service Advertising Protocol) сервера ОС NetWare. Эти пакеты каждый сервер генерирует каждую минуту, чтобы все клиенты сети могли составить правильное представление об имеющихся в сети разделяемых ресурсах — файловых службах, службах печати и т. п. SAP-пакеты распространяются в IPX-пакетах с широковещательным сетевым адресом. Удаленные мосты должны передавать широковещательные пакеты на все свои порты (маршрутизаторы не должны передавать широковещательные пакеты из сети в сеть, но для SAP-пакетов сделано исключение — маршрутизатор, поддерживающий IPX, распространяет его на все порты, кроме того, на который этот пакет поступил). Таким образом, по выделенной линии может проходить достаточно большое количество SAP-пакетов. Если эти пакеты посылаются каким-либо сервером, но не доходят до клиентов, то клиенты не могут воспользоваться службами этого сервера. Если маршрутизаторы или мосты, объединяющие сети, поддерживают технику спуфинга, то они передают по выделенному каналу не каждый SAP-пакет, а например, только каждый пятый. Интенсивность служебного трафика в глобальном канале при этом уменьшается. Но для того, чтобы клиенты не теряли из списка ресурсов удаленной сети серверы, мост (маршрутизатор) имитирует приход этих пакетов по глобальному каналу, посылая SAP-пакеты от своего имени каждую минуту, как это и положено по протоколу. При этом мост (маршрутизатор) посылает несколько раз копию реального SAP-пакета, получаемого раз в 5 минут по выделенному каналу.

Сегментация пакетов — позволяет разделять большие передаваемые пакеты и передавать их сразу через две телефонные линии. Хотя это и не делает телефонные каналы более эффективными, но все же увеличивает скорость обмена данными почти вдвое.

3.3. Коммутатор (switch)

В последнее время наблюдается вытеснение мостов коммутаторы. Коммутаторы, как и мосты работают на канальном уровне и позволяют разделить общую разделяемую среду на несколько независимых сегментов передачи данных. Алгоритм работы коммутаторов аналогичен алгоритму работы прозрачного моста. Основным отличием, обеспечившим вытеснение мостов коммутаторами – это гораздо более высокая скорость работы коммутаторов. Мост должен полностью получить кадр данных перед тем, как ретранслировать его на соответствующий порт. Коммутатор начинает ретрансляцию кадра, не дожидаясь его полного получения

(достаточно получить несколько первых байт кадра, содержащих адрес назначения). Кроме того, коммутатор позволяет организовать сразу несколько параллельных соединений между различными парами портов, что повышает пропускную способность сети в несколько раз. Однако коммутатор не может организовать одновременное соединение несколько портов – к одному порту (см. рис.).

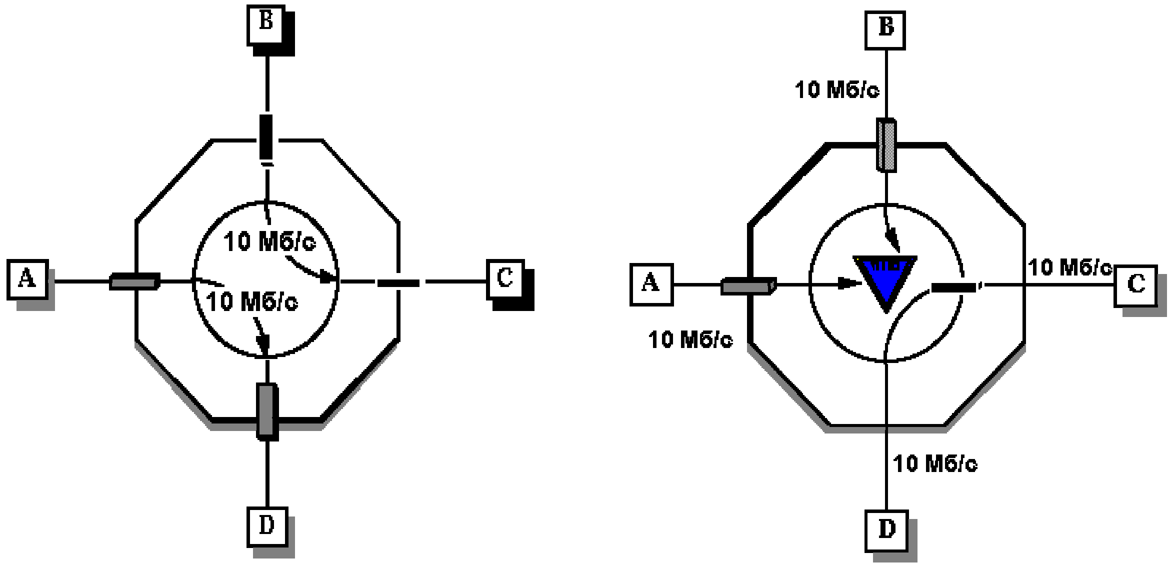
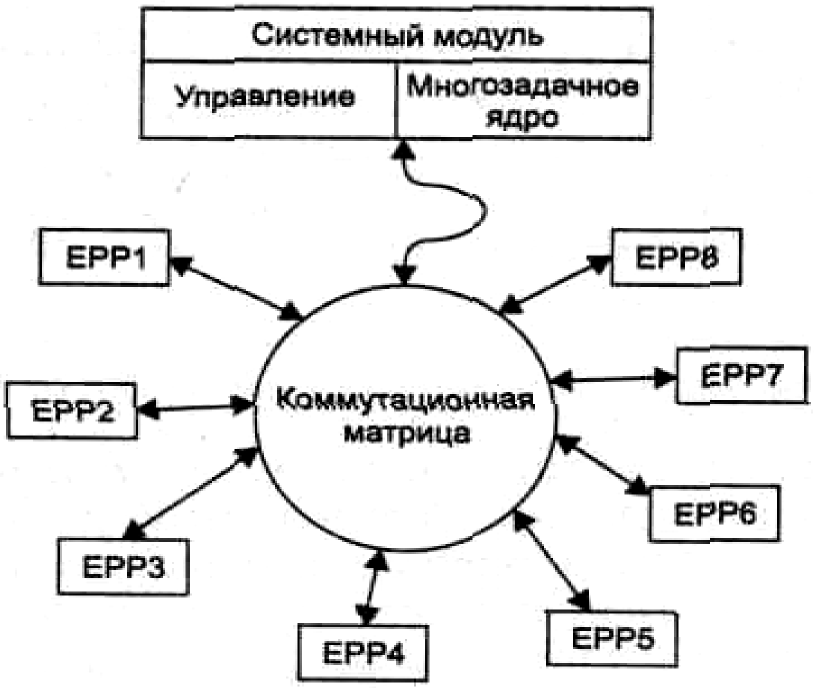


Рис. Параллельные соединения между портами коммутатора

Технология коммутаторов Ethernet была предложена фирмой Kalpana в 1990 году в ответ на растущие потребности в повышении пропускной способности сетей. Структурная схема коммутатора EtherSwitch, предложенного фирмой Kalpana, представлена ниже (см. рис.).



Каждый из 8 портов коммутатора обслуживается собственным процессором пакетов Ethernet — EPP (Ethernet Packet Processor). Кроме того, коммутатор имеет системный модуль, который координирует работу всех процессоров EPP. Системный модуль ведет общую адресную таблицу коммутатора (какие компьютеры подключены к каким портам) и обеспечивает управление коммутатором по протоколу SNMP. Для передачи кадров между портами используется коммутационная матрица, подобная тем, которые работают в телефонных коммутаторах или мультипроцессорных компьютерах. При поступлении кадра в какой-либо порт, процессор EPP буферизует несколько первых байт кадра, чтобы прочитать адрес назначения. После получения адреса назначения процессор сразу же принимает решение о передаче пакета, не дожидаясь прихода остальных байт кадра. Для этого он просматривает свой собственный кэш адресной таблицы, а если не находит там нужного адреса, обращается к системному модулю, который работает в многозадачном режиме, параллельно обслуживая запросы всех процессоров EPP. Системный модуль производит просмотр общей адресной таблицы и возвращает процессору найденную строку (адрес компьютера – номер порта), которая

запоминается процессором EPP в своем кэше для последующего использования. После определения того, к какому порту подключен сегмент компьютера – адресата, процессор EPP обращается к коммутационной матрице и пытается установить соединение с нужным портом. Если порт занят, то кадр полностью буферизуется процессором EPP входного порта, после чего процессор ожидает освобождения выходного порта. После освобождения, данные передаются на выходной порт, который получает доступ к своему сегменту сети по методу CSMA/CD и передает кадр данных в свой сегмент.

Типы коммутаторов

По конструктивному исполнению выделяют следующие типы коммутаторов:

- коммутаторы с фиксированным количеством портов
- модульные коммутаторы на основе шасси
- стековые коммутаторы
- модульно-стековые коммутаторы

Различия между этими типами коммутаторов аналогичны различиям между соответствующими типами концентраторов (см. выше).

По способу коммутации портов в коммутаторе выделяют следующие типы коммутаторов:

- коммутаторы на основе коммутационной матрицы
- коммутаторы с общей шиной
- коммутаторы с разделяемой памятью
- комбинированные коммутаторы

Коммутаторы на основе коммутационной матрицы обеспечивают основной и самый быстрый способ взаимодействия процессоров портов. Однако реализация матрицы возможна только для определенного числа портов, причем сложность схемы возрастает пропорционально квадрату количества портов коммутатора. Чисто условно коммутационную матрицу можно представить следующим рисунком:

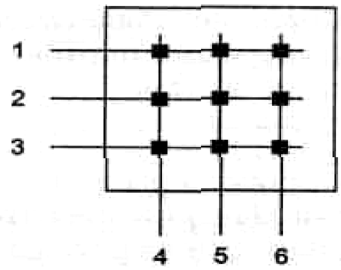


рис. Условная схема коммутационной матрицы.

Рассмотрим один из вариантов физической реализации коммутационной матрицы для 8 портов (см. рис.). Входные блоки процессоров EPP добавляют к байтам исходного кадра информацию о том на какой из портов его необходимо передать в виде специального ярлыка — тэга (tag). Для данного примера тэг представляет собой число их 3-х бит, соответствующее номеру выходного порта. Матрица состоит из трех уровней двоичных переключателей, которые соединяют свой вход с одним из двух выходов в зависимости от значения бита тэга. Переключатели первого уровня управляются первым битом тэга, второго — вторым, а третьего — третьим.

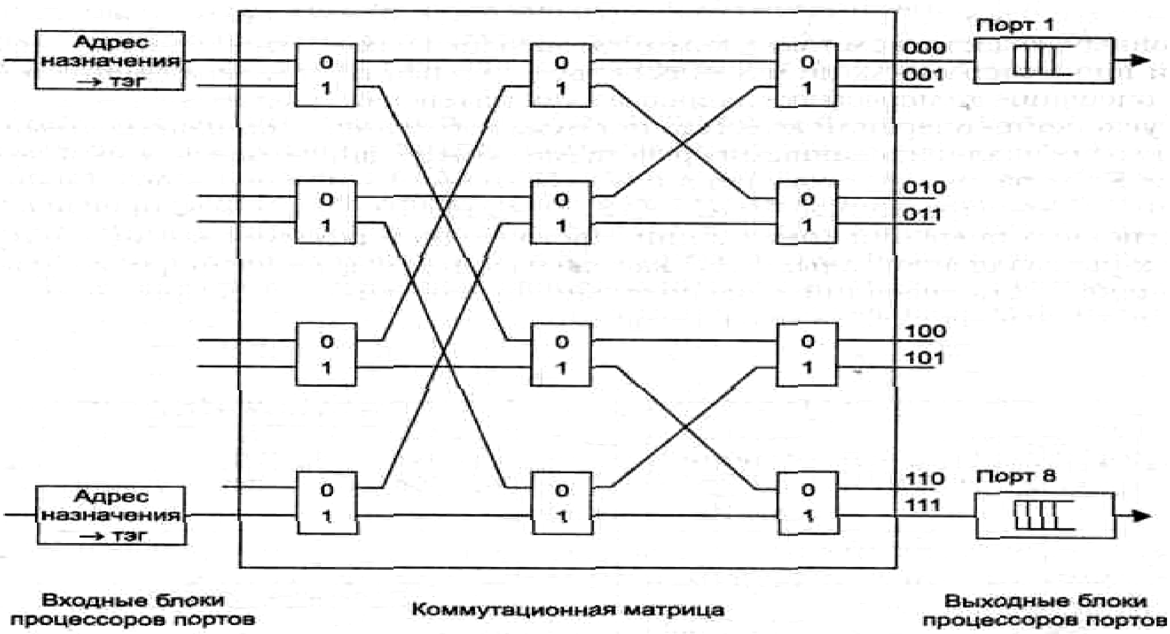


Рис. Реализация коммутационной матрицы 8x8 с помощью двоичных переключателей.

Основные достоинства таких матриц — высокая скорость коммутации портов и регулярная структура, которую удобно реализовывать в интегральных микросхемах. Недостатком является сложность наращивания числа портов и отсутствие буферизации данных внутри коммутационной матрицы (если порт занят, то данные должны накапливаться во входном блоке порта, принявшего кадр).

В коммутаторах с общей шиной процессоры портов связаны высокоскоростной шиной передачи данных, используемой в режиме разделения времени (см. рис.).

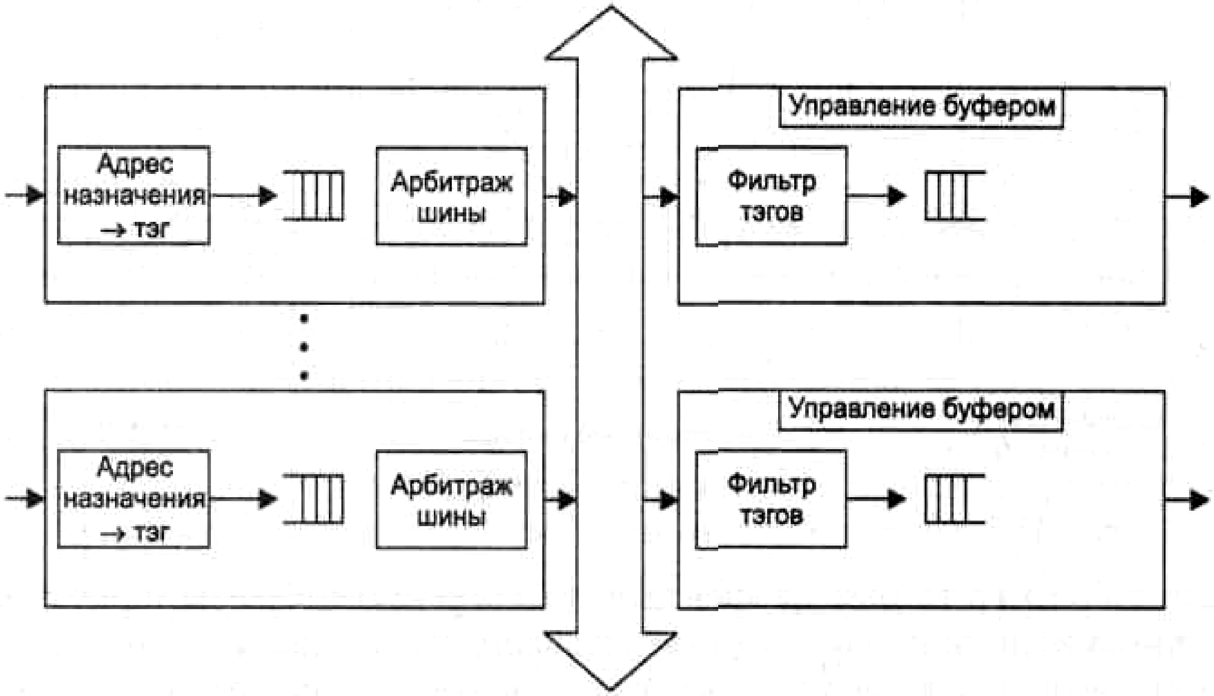


Рис. Архитектура коммутатора с общей шиной.

Каждый кадр передаваться по шине небольшими частями, по несколько байт (например, ячейками по 48 байт), чтобы обеспечить псевдопараллельную передачу кадров между несколькими портами. Входной блок процессора помещает в ячейку, переносимую по шине, тэг, в котором указывает номер порта назначения. Каждый выходной блок процессора порта содержит фильтр тэгов, который выбирает тэги, предназначенные данному порту. Достоинством коммутаторов с общей шиной является простота наращивания количества коммутируемых портов.

Коммутаторы с разделяемой памятью обеспечивают коммутацию портов при помощи общей

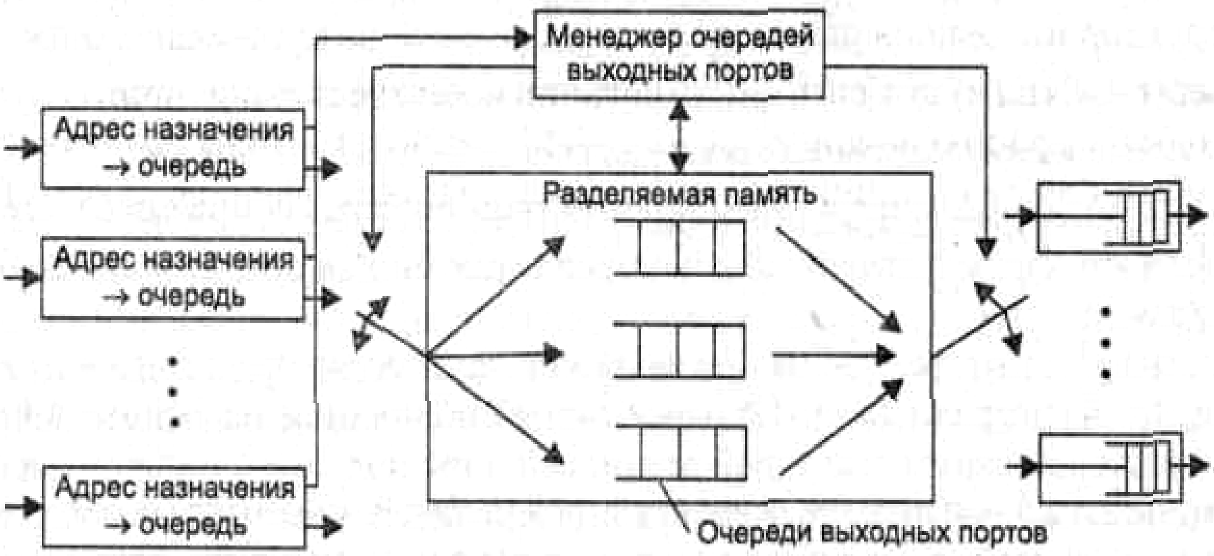


Рис. Архитектура коммутатора с общей разделяемой памятью.

Входные блоки процессоров портов соединяются с переключаемым входом разделяемой памяти, а выходные блоки этих же процессоров соединяются с переключаемым выходом этой памяти. Переключением входа и выхода разделяемой памяти управляет менеджер очередей выходных портов. В разделяемой памяти менеджер организует несколько очередей данных, по одной для каждого выходного порта. Входные блоки процессоров передают менеджеру портов запросы на запись данных в очередь того порта, который соот-

ветствует адресу назначения кадра. Менеджер по очереди подключает вход памяти к одному из входных блоков процессоров и тот переписывает часть данных кадра в очередь определенного выходного порта. По мере заполнения очередей менеджер производит также поочередное подключение выхода разделяемой памяти к выходным блокам процессоров портов, и данные из очереди переписываются в выходной буфер процессора. Достоинством коммутаров с разделяемой памятью является гибкость и экономичность распределения общей памяти между отдельными портами, что снижает требования к размеру буферной памяти процессора каждого порта.

Комбинированные коммутаторы сочетают в себе достоинства различных типов архитектур. Пример такого коммутатора, сочетающего в себе скорость матричных коммутаторов и легкость наращивания числа портов коммутаторов с общей шиной, приведен на рис. .

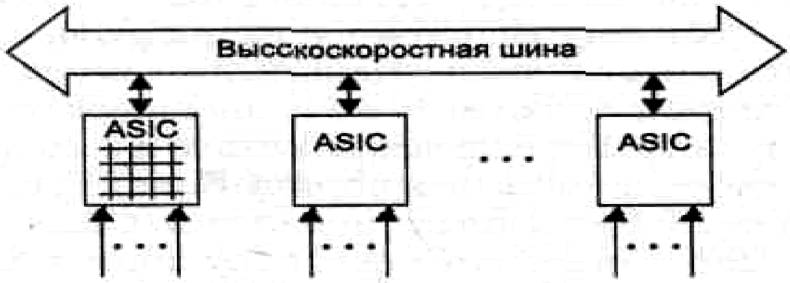


Рис. Комбинированный коммутатор.

Коммутатор состоит из модулей с фиксированным количеством портов (2-12), выполненных в виде коммутационной матрицы. Модули соединены между собой при помощи общей шины. Если порты, между которыми нужно передать кадр данных, принадлежат одному модулю, то передача кадра осуществляется при помощи коммутационной матрицы. Если же порты принадлежат разным модулям, то процессоры общаются по общей шине.

Полнодуплексный и полудуплексный режим работы коммутатора, управление потоком кадров.

Обычно к коммутатору подключаются концентраторы, т.е. на отдельный порт подключается целый сегмент. Однако к порту могут подключаться и отдельные компьютеры (микросегментация). В таком случае, коммутатор и сетевая карта компьютера могут работать в полнодуплексном режиме, т.е. одновременно передавать данные во встречных направлениях, увеличивая пропускную способность сети в два раза. Полнодуплексный режим возможен только если обе стороны - и сетевая карта и коммутатор - поддерживают этот режим. В полнодуплексном режиме не существует коллизий. Наложение двух кадров в кабеле считается нормальным явлением. Для выделения принимаемого сигнала, каждая из сторон вычитает из результирующего сигнала свой собственный сигнал.

При полудуплексном режиме работы, передача данных осуществляется только одной стороной, получающей доступ к разделяемой среде по алгоритму CSMA/CD. Полудуплексный режим фактически был подробно рассмотрен ранее.

При любом режиме работы коммутатора (полудуплексном или полнодуплексном) возникает проблема управления потоков кадров. Часто возникает ситуация, когда к одному из портов коммутатора подключен файл-сервер, к которому обращаются все остальные рабочие станции (см. рис.).

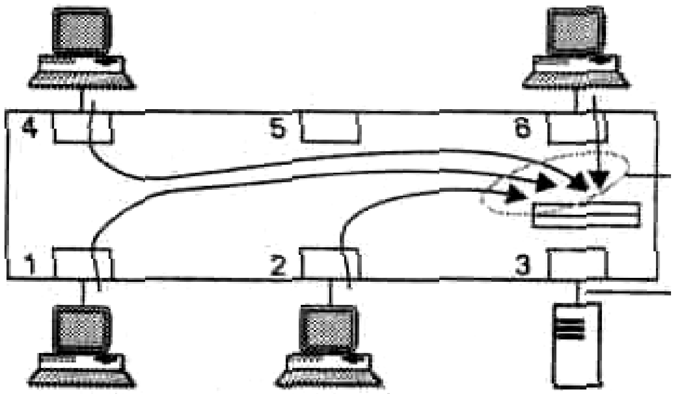


Рис. Отношение многие порты – к одному.

Если порт 3 работает на скорости 10 Мбит/с, а кадры с остальных четырех компьютеров поступают также со скоростью 10 Мбит/с, то не переданные кадры будут накапливаться в буфере порта 3 и, рано или поздно, этот буфер переполнится. Частичным решением данной проблемы было бы выделение для файл сервера порта 3, со скоростью 100 Мбит/с. Однако это не решает проблему, а лишь откладывает ее: со временем пользователи захотят более высоких скоростей работы сети, и коммутатор будет заменен на новый, у которого все порты будут работать на скорости 100 Мбит/с. Более продуманным решением, реализованном в большинстве коммутаторов, является управление потоком кадров, генерируемых компьютерами. В

полнодуплексном режиме используются специальные служебные сигналы "Приостановить передачу" и "Возобновить передачу". Получив сигнал "Приостановить передачу" сетевая карта должна прекратить передачу кадров, вплоть до последующего сигнала "Возобновить передачу" (к сожалению в текущем стандарте 802.3x не предусмотрено частичное уменьшение интенсивности передачи кадров, возможен только полный запрет). В полудуплексном режиме используется "метод обратного давления" (backpressure) и "агрессивное поведение порта коммутатора". Оба метода позволяют реализовать достаточно тонкие механизмы управления потоком кадров, частично снижая их интенсивность, но не уменьшая ее до нуля.

Метод обратного давления (backpressure) состоит в создании искусственных коллизий в сегменте, который чересчур интенсивно посылает кадры в коммутатор. Для этого коммутатор обычно использует jam-последовательность (сигналы-помехи создающие и усиливающие коллизию), отправляемую на выход порта, к которому подключен сегмент (или компьютер), чтобы приостановить его активность.

Метод агрессивного поведения порта коммутатора основан на захвате среды либо после окончания передачи очередного пакета, либо после коллизии. В первом случае коммутатор оканчивает передачу очередного кадра и, вместо технологической паузы в 9,6 мкс, делает паузу в 9,1 мкс и начинает передачу нового кадра. Компьютер не сможет захватить среду, так как он выдержал стандартную паузу в 9,6 мкс и обнаружил после этого, что среда уже занята. Во втором случае кадры коммутатора и компьютера сталкиваются и фиксируется коллизия. Компьютер делает паузу после коллизии в 51,2 мкс, как это положено по стандарту, а коммутатор — 50 мкс. И в этом случае компьютеру не удается передать свой кадр. Коммутатор может пользоваться этим механизмом адаптивно, увеличивая степень своей агрессивности по мере необходимости.

Дополнительные возможности коммутаторов

Так как коммутатор представляет собой сложное вычислительное устройство, имеющее несколько процессорных модулей, то естественно нагрузить его помимо выполнения основной функции передачи кадров с порта на порт и некоторыми дополнительными функциями. Ниже описываются наиболее распространенные дополнительные функции коммутаторов.

1) Поддержка алгоритма Spanning Tree.

Как уже отмечалось, для нормальной работы коммутатора (моста) требуется отсутствие петлевых маршрутов в сети. Петлевые маршруты могут создаваться администратором специально, для образования резервных связей, или же возникать случайным образом, что вполне возможно, если сеть имеет сложную топологию связи и плохо структурирована или документирована. Алгоритм покрывающего дерева — Spanning Tree Algorithm (STA) позволяет коммутаторам автоматически, при помощи обмена служебными пакетами, определять древовидную (без петель) конфигурацию связей в сети. В случае отказе какого-либо кабеля, порта или коммутатора, отказ обнаруживается автоматически, за счет постоянного тестирования связности сети служебными пакетами. После обнаружения потери связности протокол строит новое покрывающее дерево, если это возможно, и сеть автоматически восстанавливает работоспособность.

2) Трансляция протоколов канального уровня.

Коммутаторы позволяют преобразовывать кадры Ethernet в кадры FDDI, кадры Fast Ethernet в кадры Token Ring и т.п. Таким образом, если к одному порту коммутатора подсоединен сегмент FDDI, а к другому – сегмент Ethernet, то коммутатор позволит объединить эти две различные технологии канального уровня в единую сеть.

3) Фильтрация трафика.

Многие коммутаторы позволяют администраторам задавать дополнительные условия фильтрации кадров. Пользовательские фильтры предназначены для ограничения доступа определенных групп пользователей к определенным службам сети. Наиболее простыми являются пользовательские фильтры на основе MAC-адресов компьютеров. Самым простым вариантом является указание коммутатору отбрасывать кадры с определенным MAC-адресом. При этом пользователю, работающему на компьютере с данным MAC-адресом, полностью запрещается доступ к ресурсам другого сегмента сети. Часто администратору требуется задать более тонкие условия фильтрации, например запретить некоторому пользователю печатать свои документы на определенном сервере печати NetWare чужого сегмента, а остальные ресурсы этого сегмента сделать доступными. Для реализации такого фильтра нужно запретить передачу кадров с определенным MAC-адресом, в которых вложены пакеты IPX, в поле "номер сокета" которых будет указано значение, соответствующее службе печати NetWare. Коммутаторы не анализируют протоколы верхних уровней, поэтому администратору прейдется вручную, в шестнадцатеричной (двоичной) форме, задать такой фильтр и указать смещение и размер фильтра, относительно начала поля данных кадра канального уровня.

4) Приоритетная обработка кадров.

Использование коммутаторов позволяет реализовать приоритетную обработку кадров. Коммутатор обычно ведет для каждого входного и выходного порта не одну, а несколько очередей, причем каждая очередь имеет свой приоритет обработки. При этом коммутатор может быть сконфигурирован, например, так, чтобы передавать один низкоприоритетный пакет на каждые 10 высокоприоритетных пакетов. Основным вопросом при приоритетной обработке кадров является вопрос назначения кадру приоритета. Так как не все протоколы канального уровня поддерживают поле приоритета кадра (например, у кадров Ethernet оно отсутствует), то коммутатор должен использовать какой-либо дополнительный принцип для назначения приоритета кадру.

Наиболее распространенный способ — приписывание приоритета портам коммутатора: кадр получает соответствующий приоритет в зависимости от того, через какой порт он поступил в коммутатор. Способ несложный, но недостаточно гибкий — если к порту коммутатора подключен не отдельный компьютер, а сегмент, то все узлы сегмента получают одинаковый приоритет. Более гибким способом является использование стандарта IEEE 802.1p: в кадре Ethernet, перед полем данных, предусматривается дополнительный заголовок, состоящий из двух байт, 3 бита из которых используются для указания приоритета кадра. При передаче кадра, компьютер, при помощи специального протокола, может запросить у коммутатора один из восьми уровней приоритета кадра. Установленный коммутатором приоритет, помещается в заголовок кадра и действует для всех коммутаторов в сети. При передаче кадра сетевой карте, не поддерживающей стандарт 802.1p, дополнительный заголовок, указывающий на приоритет кадра, должен быть удален.

5) Виртуальные локальные сети (Virtual LAN, VLAN).

Коммутаторы позволяют реализовывать технологии виртуальных локальных сетей. Несмотря на схожесть терминов, не следует путать виртуальные частные сети (VPN – Virtual Private Network) и виртуальные локальные сети (Virtual LAN, VLAN). Виртуальные частные сети позволяют на сетевом уровне безопасно объединить через глобальные сети (например, Internet) или линии телефонной связи несколько локальных сетей, в единую виртуальную ЛВС. Виртуальные локальные сети позволяют на канальном уровне выделить внутри одной, физически существующей ЛВС, несколько изолированных друг от друга виртуальных ЛВС.

Виртуальной сетью называется группа узлов сети, кадры которых, в том числе и широковещательные, на канальном уровне полностью изолированы от других узлов сети (см. рис.).

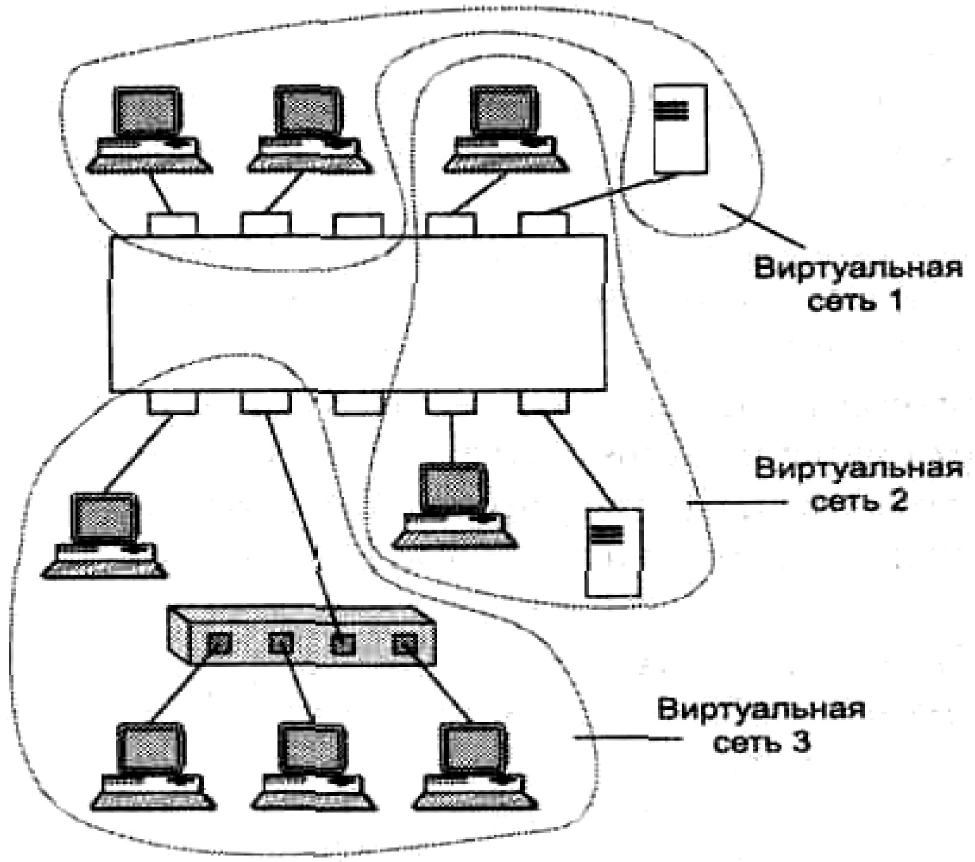


Рис. Виртуальная локальная сеть

Несмотря на то, что все узлы сети могут, например, быть подключены к одному и тому же коммутатору, передача кадров на канальном уровне между ними невозможна. Передача кадров между разными виртуальными сетями возможно только на сетевом уровне, при помощи маршрутизатора. В то же время, внутри виртуальной сети кадры передаются коммутатором стандартным образом, на канальном уровне и только на тот порт, который необходимо. Виртуальные сети могут пересекаться, если один или несколько компьютеров входят в состав более чем одной виртуальной сети (см. рис.). В таком случае, виртуальные локальные сети могут взаимодействовать между собой через эти общие компьютеры, которые могут являться, например, файловыми или почтовыми серверами.

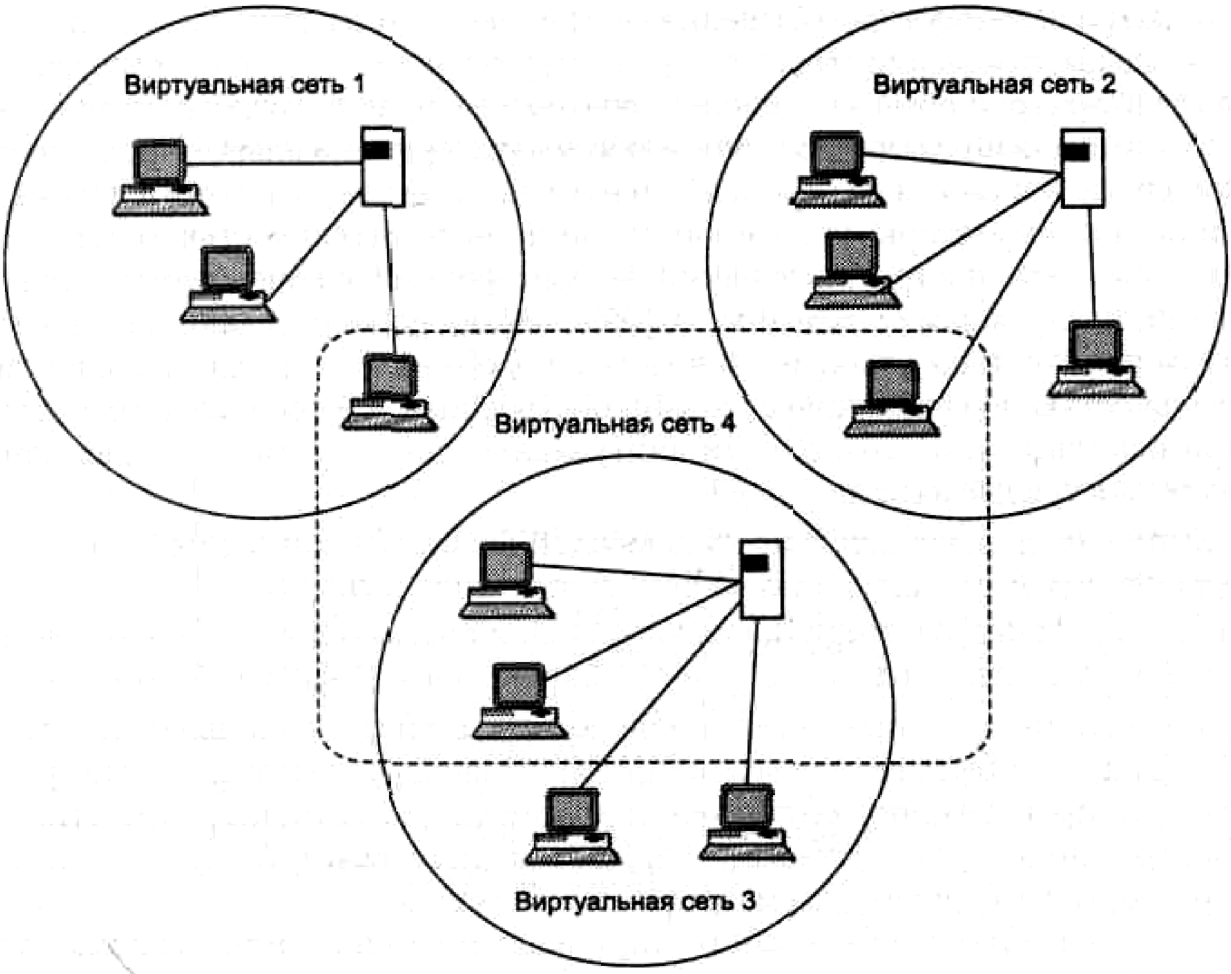


Рис. Пересечение виртуальных локальных сетей.

Технология виртуальных сетей создает гибкую основу для построения крупной сети, соединенной маршрутизаторами, так как коммутаторы позволяют создавать полностью изолированные сегменты программным путем, что очень удобно в крупных сетях. До появления технологии VLAN, для создания отдельной сети использовались либо физически изолированные сегменты коаксиального кабеля, либо несвязанные между собой сегменты, построенные на повторителях и мостах. Затем эти сети связывались маршрутизаторами в единую составную сеть (см. рис.).

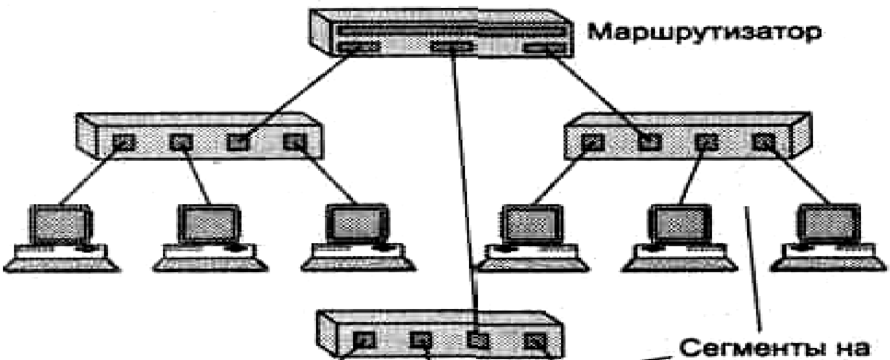


Рис. Сеть, состоящая из физически независимых подсетей.

Любое изменение структуры такой сети означало физические изменения в подключении того или иного оборудования к портам концентраторов, коммутаторов и маршрутизаторов, изменения в прокладке кабеля и т.д. В больших сетях это требовало значительных объемов работ, что повышало вероятность ошибок. При создании виртуальных сетей программным способом, порты коммутатора при помощи графической программы легко группируются в отдельные виртуальные сети. Другим, более гибким способом, является группировка в виртуальные сети не портов коммутатора, а MAC-адресов отдельных компьютеров.

Характеристики, влияющие на производительность коммутаторов

При выборе коммутатора следует в первую очередь обращать внимание на характеристики, обеспечивающие его производительность, т.к. именно это свойство послужило причиной вытеснения мостов коммутаторами. Ниже приведены некоторые характеристики:

1) Скорость фильтрации/продвижения кадров (кадров в секунду), пропускная способность (мегабит в секунду), задержка передачи кадра.

Коммутатор является *неблокирующим*, если он может передавать кадры через свои порты с той же скоростью, с какой они на них поступают. Необходимо учитывать для кадров какого протокола и какой длины указана пропускная способность. Максимальная пропускная способность всегда достигается на кадрах максимальной длины, так как при этом доля накладных расходов на служебную информацию кадра гораздо ниже, чем для кадров минимальной длины, а время обработки кадра (в расчете на один байт полезной информации), существенно меньше. Поэтому коммутатор может быть блокирующим для кадров минимальной длины, но при этом иметь очень хорошие показатели пропускной способности. Коммутатор – это многопортовое устройство, поэтому для него все приведенные выше характеристики (кроме задержки передачи кадра) можно давать в общей сумме, или в расчете на один порт. Обычно производители коммутаторов указывают общую максимальную пропускную способность устройства.

2) Тип коммутации — "на лету" или с полной буферизацией.

При коммутации на лету передача кадра начинается сразу после приема первых нескольких байт заголовка. При коммутации с полной буферизацией кадр должен быть полностью принят в буферную память до начала передачи. Разницу между этими характеристиками коммутаторов иллюстрирует следующая таблица:

Таблица

Возможности коммутаторов при коммутации "на лету" и с полной буферизацией.

Функция	Налету	С буферизацией
Защита от плохих кадров	Нет	Да
Трансляция протоколов разнородных сетей (Ethernet Token Ring, FDDI, ATM)	Нет	Да
Задержка передачи пакетов	Низкая (5-40 мкс) при низкой нагрузке, средняя при высокой нагрузке	Средняя при любой нагрузке
Поддержка резервных связей	Нет	Да
Функция анализа трафика	Нет	Да

Так как каждый способ имеет свои достоинства и недостатки, в тех моделях коммутаторов, которым не нужно транслировать протоколы, иногда применяется механизм адаптивной смены режима работы коммутатора. Основной режим такого коммутатора — коммутация "на лету", но коммутатор постоянно контролирует трафик и при превышении интенсивности появления плохих кадров некоторого порога переходит на режим полной буферизации. Затем коммутатор может вернуться к коммутации "на лету".

3) Размер адресной таблицы.

Максимальная емкость адресной таблицы определяет предельное количество MAC-адресов, с которыми может одновременно оперировать коммутатор. Так как коммутаторы чаще всего используют для выполнения операций каждого порта выделенный процессорный блок со своей памятью для хранения экземпляра адресной таблицы, то размер адресной таблицы для коммутаторов обычно приводится в расчете на один порт. Каждый порт хранит только те наборы адресов, с которыми он работал в последнее время. Недостаточная емкость адресной таблицы может служить причиной замедления работы коммутатора и засорения сети избыточным трафиком. Если адресная таблица процессора порта полностью заполнена, а он встречает новый адрес источника в поступившем пакете, процессор должен вытеснить из таблицы какой-либо старый адрес и поместить на его место новый. Эта операция сама по себе отнимет у процессора часть времени, но главные потери производительности будут наблюдаться при поступлении кадра с адресом назначения, который пришлось удалить из адресной таблицы. Так как адрес назначения кадра неизвестен, то коммутатор должен передать этот кадр на все остальные порты. Эта операция будет создавать лишнюю работу для многих процессоров портов, кроме того, копии этого кадра будут попадать и на те сегменты сети, где они совсем не обязательны. Некоторые производители коммутаторов решают эту проблему за счет того, что один из портов коммутатора конфигурируется как магистральный порт, на который по умолчанию передаются все кадры с неизвестным адресом. Передача кадра на магистральный порт производится в расчете на то, что этот порт подключен к вышестоящему коммутатору, который имеет большую емкость адресной таблицы и знает, куда нужно передать любой кадр.

4) Объем буфера кадров.

Внутренняя буферная память коммутатора нужна для временного хранения кадров данных в тех случаях, когда их невозможно немедленно передать на выходной порт. Буфер предназначен для сглаживания кратковременных пульсаций нагрузки на сеть. Каждый процессорный модуль порта обычно имеет свою буферную память. Чем больше объем этой памяти, тем менее вероятны потери кадров при перегрузках. Обычно коммутаторы, предназначенные для работы в ответственных частях сети, имеют буферную память в несколько десятков или сотен килобайт на порт. Хорошо, когда эту буферную память можно перераспределять между несколькими портами, так как одновременные перегрузки по нескольким портам маловероятны. Дополни-

тельным средством защиты может служить общий для всех портов буфер в модуле управления коммутатором. Такой буфер обычно имеет объем в несколько мегабайт.

5) Производительность процессоров портов, производительность внутренней шины коммутатора. Необходимо следить, чтобы производительность общей шины (при архитектуре с общей шиной) или производительность процессоров портов не стала "узким местом" в коммутаторе.

3.4. Маршрутизатор (router)

Маршрутизаторы необходимы в крупных сетях, для объединения сегментов, построенных на концентраторах, мостах и коммутаторах. Маршрутизатор может быть реализован в виде отдельного высокопроизводительного устройства (например, маршрутизаторы компании Cisco Systems), или функцию маршрутизации может выполнять сетевая операционная система обычного компьютера, подключенного одновременно к нескольким сетям, при помощи нескольких сетевых карт (шлюз). Маршрутизаторы работают на сетевом уровне модели OSI и не накладывают ограничений на топологию сети. Если для мостов и коммутаторов обязательно отсутствие петлевых маршрутов в сети (древовидная структура), то маршрутизатор работает в сетях с произвольной топологией и обеспечивает выбор оптимального маршрута для доставки пакетов. Использование древовидной структуры для крупных сетей нерационально, т.к. в таком случае на корневой коммутатор (мост) приходится слишком большая нагрузка, а его отказ приводит к распадению сети на отдельные фрагменты и потере пользователями доступа к большому количеству ресурсов сети. Поэтому рационально строить сети по децентрализованному принципу, когда между любыми двумя компьютерами может существовать множество маршрутов. Именно нахождением и ведением таблицы таких маршрутов (таблицы маршрутизации) и доставкой пакетов по оптимальному маршруту занимается маршрутизатор.

Другой функцией маршрутизаторов является объединение в единую сеть сегментов, работающих на различных протоколах канального уровня. Например, объединение сегментов Fast Ethernet и FDDI. Маршрутизатор работает на сетевом уровне модели OSI (например, по протоколу IP), и для него не существенно какие протоколы канального уровня используются в сегментах. Трансляция протоколов (кадры Fast Ethernet в кадры FDDI) может осуществляться и некоторыми моделями коммутаторов, однако такая возможность появилась сравнительно недавно, и исторически для объединения разнородных сетей используют маршрутизаторы. Кроме того, коммутаторы в некоторых случаях не могут корректно выполнить трансляцию кадров. Например, коммутаторами не поддерживается функция фрагментации кадров и, если в объединяемых сетях не совпадают максимально допустимые размеры кадров, то коммутатор не сможет транслировать очень большие кадры.

Сегодня считается, что любая крупная сеть должна включать изолированные сегменты, соединенные маршрутизаторами, иначе потоки ошибочных кадров, например широковещательных, будут периодически затапливать всю сеть через прозрачные для них коммутаторы (мосты), приводя ее в неработоспособное состояние. Кроме того, использование маршрутизаторов позволяет структурировать сеть (подсеть отдела кадров, подсеть бухгалтерии и т.п.) и легче реализовывать политику безопасности, за счет использования межсетевых экранов. Межсетевой экран (firewall, брандмауэр) – это специальное программное обеспечение, которое установлено на маршрутизаторе, или компьютере-шлюзе, выполняющем функции маршрутизатора, и позволяющее контролировать доступ пользователей к тем или иным ресурсам сети. Для межсетевого экрана задаются правила фильтрации вида: "через межсетевой экран допускается прохождение пакетов с IP-адресом отправителя 172.18.10.1 (порт 80) и IP-адресом получателя 192.168.1.1 (порт 21), в четверг с 15.00 до 19.00". Пакеты, не удовлетворяющие правилам фильтрации отбрасываются, а факт их наличия регистрируется в специальном журнале.

В крупных сетях выбор наилучшего маршрута часто является достаточно сложной задачей, с математической точки зрения. Особенно интенсивных вычислений требуют протоколы OSPF, NLSP, IS-IS, вычисляющие оптимальный путь на графе. Кроме того маршрутизатор вынужден выполнять буферизацию, фильтрацию, фрагментацию пакетов и другие задачи. При этом очень важна производительность маршрутизатора, поэтому типичный маршрутизатор крупных сетей является мощным вычислительным устройством с одним или даже несколькими процессорами (часто специализированными или построенными на RISC-архитектуре) и сложным программным обеспечением, работающим под управлением специализированной операционной системы реального времени. Многие разработчики маршрутизаторов построили в свое время такие операционные системы на базе ОС Unix, естественно, значительно ее переработав.

Алгоритмы маршрутизации

Маршрутизация – это выбор маршрута доставки пакета. Частично, алгоритм маршрутизации уже был рассмотрен в лекции, посвященной протоколу IP. Здесь будет рассмотрена только общая классификация алгоритмов маршрутизации, не упоминавшихся ранее.

По степени распределенности схемы маршрутизации выделяют следующие алгоритмы маршрутизации:

- Одношаговые алгоритмы. Наиболее широко распространены в сетях. В таблице маршрутизации хранится информация только об одном шаге маршрута (ближайший маршрутизатор на пути к адресату). При отсутствии возможности доставки пакета напрямую (когда маршрутизатор различными сетевыми интерфейсами одновременно подключен и к сети отправителя и к сети адресата), пакет

доставляется на следующий ближайший маршрутизатор на пути к адресату, который анализирует свои таблицы маршрутизации и занимается дальнейшей доставкой пакета. Подробнее см. лекции по протоколу IP.

- Многошаговые алгоритмы. В таблице маршрутизации указываются все шаги маршрута (промежуточные маршрутизаторы), которые должен пройти пакет. Схема работы - аналогично мостам, с маршрутизацией от источника (см. ранее). В сетях распространена мало. Однако в новой версии протокола IP (IPv6), наряду с классической одношаговой маршрутизацией, будет разрешена и маршрутизация от источника.

По способу построения таблиц маршрутизации выделяют следующие алгоритмы маршрутизации:

- Алгоритмы статической маршрутизации. Все записи в таблице маршрутизации задаются администратором вручную. Пригоден только для небольших сетей. В крупных сетях применяется только совместно с алгоритмом динамической маршрутизации.
- Алгоритмы динамической маршрутизации (таблицы маршрутизации составляются и обновляются автоматически, на основании имеющейся информации о непосредственно подключенных к маршрутизатору сетях и информации от соседних маршрутизаторов, передаваемой по протоколам RIP, OSPF, NLSP, подробнее см. ниже).
- Алгоритмы простой маршрутизации. В сетях практически не применяются. Используется случайная маршрутизация (прибывший пакет посылается в первом попавшемся случайном направлении, кроме исходного), лавинная маршрутизация (пакет широковещательно посылается по всем возможным направлениям, кроме исходного), маршрутизация по предыдущему опыту (выбор маршрута осуществляется аналогично выбору маршрута в прозрачных мостах).

По использованию маски подсети в процессе маршрутизации IP-пакетов выделяют:

- Маршрутизацию на основании классов IP-адресов, без использования маски подсети.
- Бесклассовая междоменная маршрутизация, с использованием маски подсети.

При маршрутизации на основании классов IP-адресов, в таблицах маршрутизации маски подсетей не хранятся. Решение о том, является ли данный IP-адрес адресом сети или адресом конкретного компьютера принимается на основании класса IP-адреса (у сетей класса С адрес компьютера находится в последнем октете, у сетей класса В адрес компьютера находится в последних двух октетах и т.д.). Такой подход прост, но создает неудобства, т.к. минимальный размер подсети составляет 253 компьютера (сеть класса С), что является нерациональным расходованием адресов и не позволяет структуризировать сеть на более мелкие подсети. Поэтому постепенно в сетях происходит переход на маршрутизацию с использованием масок подсети - бесклассовая междоменная маршрутизация (CIDR, Classless Inter-Domain Routing). При этом подходе подсетям выделяются непрерывные диапазоны IP-адресов так, чтобы номер компьютера и номер сети можно было описать при помощи маски подсети (подробнее см. лекции по протоколу IP). При обмене информацией между маршрутизаторами (например, по протоколу RIPv2), вместе с информацией о маршрутах передается и информация о масках подсетей для соответствующих IP-адресов.

Протоколы динамической маршрутизации.

Протоколы маршрутизации обеспечивают обмен служебной информацией, необходимой для построения таблиц маршрутизации. Существует множество протоколов маршрутизации, однако здесь мы рассмотрим только два протокола: RIP (представитель дистанционно-векторных протоколов) и OSPF (представитель протоколов состояния связей).

Построение таблицы маршрутизации по протоколу RIP (Routing Information Protocol) происходит следующим образом:

1) Маршрутизаторы создают минимальные таблицы маршрутизации, на основании имеющейся информации о сетях, непосредственно подключенным к их сетевым интерфейсам.
2) Маршрутизаторы рассылают минимальные таблицы соседям (соседями считаются те маршрутизаторы, которые могут получить сообщение напрямую, не пользуясь услугами промежуточных маршрутизаторов, т.е. маршрутизаторы которые одним из своих сетевых интерфейсов подключены к той же сети, что и маршрутизатор, отправляющий сообщение).
3) Маршрутизаторы анализируют полученные минимальные таблицы других маршрутизаторов, наращивая поле "метрика" (расстояние до сети/компьютера) на единицу, учитывая таким образом тот соседний маршрутизатор, от которого была получена минимальная таблица, как еще один маршрутизатор на пути до сети назначения. После этого полученная минимальная таблица сравнивается с уже имеющейся таблицей маршрутизации. Если в обоих таблицах имеется несколько маршрутов до одной и той же сети, то в результирующую таблицу попадает вариант с наименьшей метрикой (расстоянием до сети).
4) Рассылка новой, уже не минимальной, таблицы соседям и последующая обработка полученных от соседей не минимальных таблиц. Соседям рассылается полный вариант имеющейся таблицы маршрутизации, за исключением информации о сетях, которая была получена непосредственно от самих этих соседей. Делается это для предотвращения создания петлевых маршрутов и зацикливания пакетов (техника "расщепления горизонта" – split horizont). Этап 4 повторяется циклически каждые 30 секунд (протокол RIP IP). В

результате, все маршрутизаторы в сети будут иметь корректную таблицу маршрутизации: информация из таблиц любого маршрутизатора, через соседей, рано или поздно дойдет до любого другого маршрутизатора в сети. Более того, при изменениях в сети (подключение к какому-либо маршрутизатору новой сети или временная недоступность старой сети) информация об изменениях также распространится по сети.

5) Для автоматического обновления таблиц маршрутизации, каждая запись, созданная при помощи протокола RIP, имеет свой срок жизни (TTL, Time To Live). В RIP IP срок жизни записи равен шести периодам рассылки таблиц маршрутизации, т.е. 180 секунд. Если какой-либо маршрутизатор выходит из строя и перестает слать своим соседям сообщения о сетях, которые можно достичь через него, то через 180 секунд все записи, которые породил этот маршрутизатор, станут недействительными у его ближайших соседей. После этого процесс повторится уже для соседей ближайших соседей – они вычеркнут подобные записи уже через 360 секунд, так как первые 180 секунд ближайшие соседи еще передавали сообщения об этих записях и т.д. Как видно из объяснения, сведения о недоступных через отказавший маршрутизатор сетях распространяются по сети не очень быстро. Поэтому есть более быстрый способ объявить сеть недоступной. Если отказал не весь маршрутизатор (что случается редко), а только один из его интерфейсов, подключенных к какой-либо сети, то при следующем обмене таблицами маршрутизации (через 30 секунд), маршрутизатор укажет "бесконечное" расстояние до недоступной сети, что приведет к исключению данного маршрута из таблиц других маршрутизаторов. Бесконечному расстоянию в протоколе RIP IP соответствует метрика 16 (16 маршрутизаторов между отправителем и получателем). Такое небольшое значение "бесконечного" расстояния связано с низкой скоростью распространения информации об отказах между маршрутизаторами (см. выше). Большее значение может привести к длительным периодам зацикливания и потерь пакетов.

Протокол OSPF (Open Shortest Path First) является более современным и эффективным, чем протокол RIP. В OSPF процесс построения таблицы маршрутизации происходит по следующим этапам:

1) Каждый маршрутизатор строит граф связей сети, в котором вершинами графа являются маршрутизаторы и IP-сети, а ребрами — интерфейсы маршрутизаторов. Все маршрутизаторы для этого обмениваются со своими соседями той информацией о графе сети, которой они располагают к данному моменту времени. Этот процесс похож на процесс распространения векторов расстояний до сетей в протоколе RIP, однако сама информация качественно другая — это информация о топологии сети. Эти сообщения называются router links advertisement — объявление о связях маршрутизатора. Кроме того, при передаче топологической информации маршрутизаторы ее не модифицируют, как это делают RIP-маршрутизаторы, а передают в неизменном виде. В результате распространения топологической информации все маршрутизаторы сети располагают идентичными сведениями о графе сети, которые хранятся в топологической базе данных маршрутизатора.
2) Нахождении оптимальных маршрутов с помощью полученного графа. Каждый маршрутизатор считает себя центром сети и ищет оптимальный маршрут до каждой известной ему сети. В каждом найденном таким образом маршруте запоминается только один шаг – до следующего маршрутизатора, в соответствии с принципом одношаговой маршрутизации. Данные об этом шаге и попадают в таблицу маршрутизации. Задача нахождения оптимального пути на графе является математически достаточно сложной и трудоемкой. В протоколе OSPF для ее решения используется итеративный алгоритм Дийкстры. Если несколько маршрутов имеют одинаковую метрику до сети назначения, то в таблице маршрутизации запоминаются первые шаги всех этих маршрутов.
3) После первоначального построения таблицы маршрутизации необходимо отслеживать изменения состояния сети и вносить коррективы в таблицу маршрутизации. Для контроля состояния связей в сети OSPF-маршрутизаторы не используют обмен полной таблицей маршрутизации, как это не очень рационально делают RIP-маршрутизаторы. Вместо этого они передают специальные короткие сообщения HELLO. Если состояние сети не меняется, то OSPF-маршрутизаторы корректировкой своих таблиц маршрутизации не занимаются и не посылают соседям объявления о связях. Если же состояние связи изменилось, то ближайшим соседям посылается новое объявление, касающееся только данной связи, что экономит пропускную способность сети. Получив новое объявление об изменении состояния связи, маршрутизатор перестраивает граф сети, заново ищет оптимальные маршруты (не обязательно все, а только те, на которых отразилось данное изменение) и корректирует свою таблицу маршрутизации. Одновременно маршрутизатор ретранслирует объявление каждому из своих ближайших соседей (кроме того, от которого он получил это объявление).

Автономные системы, протоколы внешней маршрутизации.

Протоколы RIP и OSPF используются внутри локальных сетей, или, если рассматривать сеть Internet, внутри автономных систем.

<u>Автономная система</u> – это объединение локальных сетей с одинаковой маршрутной политикой и общей администрацией, например совокупность сетей компании Ростелеком. Каждая автономная система регистрируется (за довольно существенную плату) в региональной регистратуре Internet. Для Европы и России – это RIPE (http://www.ripe.net). Если Вам известен IP-адрес, то используя программу ***<u>WhoIS</u>*** (входит во все дистрибутивы Linux/Unix, или, например, в программу IPTOOLS для Windows) можно обратиться в

одну из региональных регистратур (whois.ripe.net, whois.internic.net, whois.ripn.net, whois.arin.net, whois.apnic.net, whois.nic.mil, whois.nic.gov) и получить сведения об автономной системе, за которой числится данный IP-адрес: адрес организации, имя ответственного, телефоны и т.д.

Маршрутизация между автономными системами осуществляется пограничными маршрутизаторами (border gateways). При маршрутизации используются протоколы внешней маршрутизации, в частности BGP (Border Gateway Protocol). Его принципиальным отличием от протоколов внутренней маршрутизации (RIP, OSPF) является наличие маршрутной политики. Маршрутная политика позволяет передавать другим погарничным маршрутизаторам не все существующие маршруты, а только те, которые администрация автономной системы сочтет нужными. Также, для различных маршрутов можно задавать дополнительные параметры, характеризующие пропускную способность маршрута, стоимость транзита трафика по данному маршруту и т.д. Приведем пример (см. рис.).

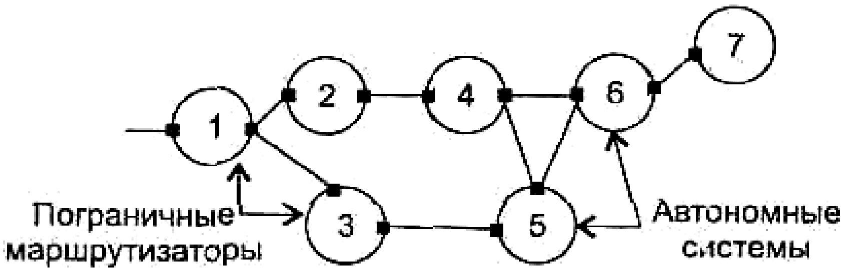


рис. Использование протокола BGP.

Пример 1

Администрация автономной системы 2 (АС2) не хочет, чтобы сетевой трафик из автономной системы 1 (АС1) проходил транзитом через нее. Поэтому, хотя автономные системы АС4, АС5, АС6, АС7, АС3 и доступны для АС1 через АС2, однако АС2 не объявляеет об этом. Такое возможно только при ручном составлении таблиц маршрутизации или использовании протокола BGP (протоколы RIP и OSPF не позволяют этого сделать).

Пример 2

Кратчайший маршрут из АС1 в АС5 лежит через АС3. Однако стоимость транзита через АС3 для АС1 чрезвычайно велика. Другой маршрут АС1–АС2–АС4–АС5 короче маршрута АС1–АС2–АС4–АС6–АС5, однако связь АС4-АС5 значительно "медленне" (33,6 Кбит/с), чем связь АС4–АС6–АС5 (2,488 Гбит/с), поэтому маршрут АС1–АС2–АС4–АС6–АС5 предпочтительнее. Протокол BGP, при соответствующей настройке пограничных маршрутизаторов, позволяеет учесть все эти особенности.

Классификация маршрутизаторов.

По областям применения маршрутизаторы делятся на несколько классов.

Магистральные маршрутизаторы (backbone routers) предназначены для построения главной магистрали сети. Это наиболее мощные устройства, способные обрабатывать несколько сотен тысяч или даже несколько миллионов пакетов в секунду, имеющие большое количество интерфейсов локальных (Ethernet, Token Ring, FDDI) и глобальных сетей (T1/E1, SDH, ATM). Чаще всего магистральный маршрутизатор выполнен по модульной схеме на основе шасси, с большим количеством слотов (≈12). Большое внимание уделяется надежности и отказоустойчивости маршрутизатора, которая достигается за счет системы терморегуляции, избыточных источников питания, заменяемых "на ходу" модулей. Примерами магистральных маршрутизаторов могут служить маршрутизаторы Backbone Concentrator Node (BCN) компании Nortel Networks (ранее Bay Networks), Cisco 7500, Cisco 12000.

Маршрутизаторы региональных отделений соединяют региональные отделения между собой и с центральной сетью. Такой маршрутизатор обычно представляет собой упрощенную версию магистрального маршрутизатора. Если он выполнен на основе шасси, то количество слотов его шасси меньше (≈4-5). Возможна также реализация с фиксированным числом портов. Меньше список поддерживаемых интерфейсов локальных и глобальных сетей. Примерами маршрутизаторов региональных отделений могут служить маршрутизаторы BLN, ASN компании Nortel Networks, Cisco 3600, Cisco 2500, NetBuilder II компании 3Com.

Маршрутизаторы удаленных офисов соединяют, как правило, единственную локальную сеть удаленного офиса с центральной сетью предприятия по глобальной связи. Обычно такие маршрутизаторы поддерживают для локальной сети интерфейс Ethernet 10 Мбит/с (Fast Ethernet 100 Мбит/с), а для глобальной сети — выделенную линия со скоростью 64 Кбит/с, 1544 Кбит/с или 2 Мбит/с (может также поддерживаться коммутируемая телефонная линия, в качестве резервной связи). Представителями этого класса являются маршрутизаторы Nautika компании Nortel Networks, Cisco 1600, Office Connect компании 3Com, семейство Pipeline компании Ascend.

Маршрутизаторы локальных сетей (коммутаторы 3-го уровня) предназначены для разделения крупных локальных сетей на подсети. Основное требование, предъявляемое к ним, — высокая скорость маршрутизации. В коммутаторах 3-го уровня отсутствуют низкоскоростные порты (такие как модемные порты 33,6 Кбит/с). Все порты имеют скорость по крайней мере 10 Мбит/с, а многие работают на скорости

100 Мбит/с. Примерами коммутаторов 3-го уровня служат коммутаторы CoreBuilder 3500 компании 3Com, Accelar 1200 компании Nortel Networks, Waveswitch 9000 компании Plaintree, Turboiron Switching Router компании Foudry Networks. Подробнее коммутаторы третьего уровня будут рассмотрены ниже.

Дополнительные функции маршрутизаторов.

1) Поддержка нескольких сетевых протоколов. Приоритеты сетевых протоколов.

Маршрутизатор, поддерживающий несколько протоколов сетевого уровня (например, IP и IPX), называют многопротокольным. Можно установить приоритет одного протокола сетевого уровня над другими. На выбор маршрутов эти приоритеты не оказывают никакого влияния, они влияют только на порядок, в котором многопротокольный маршрутизатор обслуживает пакеты разных сетевых протоколов.

2) Поддержка одновременно нескольких протоколов маршрутизации.

В протоколах маршрутизации обычно предполагается, что маршрутизатор автоматически строит свою таблицу на основе работы только этого одного протокола маршрутизации (например, только протокола RIP). Тем не менее, иногда в большой сети приходится поддерживать одновременно несколько протоколов маршрутизации, чаще всего это складывается исторически. При этом таблица маршрутизации может получаться противоречивой – разные протоколы маршрутизации могут выбрать разные следующие маршрутизаторы для какой-либо сети назначения. Большинство маршрутизаторов решает эту проблему за счет придания приоритетов решениям разных протоколов маршрутизации. Высший приоритет отдается статическим маршрутам (администратор всегда прав), следующий приоритет имеют маршруты, выбранные протоколами состояния связей (OSPF или NLSP), а низшим приоритетов обладают маршруты дистанционно-векторных протоколов (RIP), как самых несовершенных.

3) Поддержка политики маршрутных объявлений.

В большинстве протоколов обмена маршрутной информации (RIP, OSPF, NLSP) предполагается, что маршрутизатор объявляет в своих сообщениях обо всех сетях, которые ему известны. Аналогично предполагается, что маршрутизатор при построении своей таблицы учитывает все адреса сетей, которые поступают ему от других маршрутизаторов сети. Однако существуют ситуации, когда администратор хотел бы скрыть существование некоторых сетей от других администраторов, например, по соображениям безопасности. Или же администратор хотел бы запретить некоторые маршруты, которые могли бы существовать в сети. При статическом построении таблиц маршрутизации решение таких проблем не составляет труда. Динамические же протоколы маршрутизации не позволяют стандартным способом реализовывать подобные ограничения. Существует только один широко используемый протокол динамической маршрутизации, в котором описана возможность существования правил (policy), ограничивающих распространение некоторых адресов в объявлениях, — это протокол BGP. Разработчики маршрутизаторов исправляют недостатки протоколов RIP, OSPF и NLSP, вводя в маршрутизаторы поддержку правил передачи и использования маршрутной информации, подобных тем, которые рекомендует BGP.

4) Поддержка немаршрутизируемых протоколов.

Немаршрутизируемые протоколы, такие как NetBIOS (NetBEUI), не работают с адресами сетевого уровня. Маршрутизаторы могут обрабатывать пакеты таких протоколов двумя способами. В первом случае они работают с пакетами этих протоколов на канальном уровне, как прозрачные мосты. Маршрутизатор необходимо сконфигурировать так, чтобы по отношению к немаршрутизируемым протоколам на некоторых портах он выполнял функции моста, а по отношению к маршрутизируемым протоколам — функции маршрутизатора. Такой маршрутизатор иногда называют brouter (bridge + router). Другим способом передачи пакетов немаршрутизируемых протоколов является инкапсуляция этих пакетов в пакеты протокола сетевого уровня, чаще всего в IP-пакеты. Некоторые производители маршрутизаторов разработали собственные протоколы, специально предназначенные для инкапсуляции немаршрутизируемых пакетов.

5) Разделение функций построения и использования таблицы маршрутизации.

Основная вычислительная работа проводится маршрутизатором при составлении таблицы маршрутизации с маршрутами ко всем известным ему сетям. Эта работа состоит в обмене пакетами протоколов маршрутизации (RIP, OSPF, NLSP) и вычислении оптимального пути к каждой целевой сети по некоторому критерию. Для вычисления оптимального пути на графе, как того требуют протоколы OSPF и NLSP, необходимы значительные вычислительные мощности. После того как таблица маршрутизации составлена, продвижение пакетов происходит весьма просто, на основании просмотра таблицы маршрутизации. Некоторые маршрутизаторы поддерживают только функции продвижения пакетов по готовой таблице маршрутизации. Такие маршрутизаторы являются усеченными маршрутизаторами, так как для их полноценной работы требуется наличие обычного маршрутизатора, у которого можно взять готовую таблицу маршрутизации. Этот маршрутизатор часто называется сервером маршрутов. Отказ от самостоятельного построения таблицы маршрутизации резко удешевляет маршрутизатор и повышает его производительность. Примерами такого подхода являются маршрутизаторы NetBuilder компании 3Com, поддерживающие фирменную технологию Boundary Routing, маршрутизирующие коммутаторы Catalyst 5000 компании Cisco Systems.

Основные технические характеристики маршрутизатора.

1) Перечень поддерживаемых сетевых протоколов и протоколов машрутизации.

При выборе маршрутизатора необходимо учитывать какие протоколы (в том числе и устаревшие) используются или будут использоваться на предприятии. Перечень поддерживаемых сетевых протоколов обычно включает протоколы IP, IPX, AppleTalk, CONS и CLNS OSI, DECnet, Banyan VINES, Xerox XNS. Перечень протоколов маршрутизации составляют протоколы: IP RIP, IPX RIP, NLSP, OSPF, IS-IS OSI, EGP, BGP, VINES RTP, AppleTalk RTMP.

2) Перечень поддерживаемых интерфейсов локальных и глобальных сетей.

Маршрутизатор должен иметь интерфейсы ко всем протоколам канального уровня, использующимся в локальной сети предприятия, а также для связи с глобальными сетями. Для локальных сетей – это интерфейсы Ethernet, Token Ring, FDDI, Fast Ethernet, Gigabit Ethernet, 100VG-AnyLAN. Для глобальных связей – это интерфейсы V.21-V.90 (модем), T1/E1, T3/E3, SONET/SDH, ISDN, интерфейсы к сетям X.25, Frame Relay, ATM, а также поддержка протокола канального уровня PPP.

3) Общая производительность маршрутизатора.

Общая производительность маршрутизаторов колеблется от нескольких десятков тысяч пакетов в секунду до нескольких миллионов пакетов в секунду и зависит от многих факторов, наиболее важными из которых являются: тип используемых процессоров, эффективность программной реализации протоколов, архитектурная организация вычислительных и интерфейсных модулей. Наиболее производительные маршрутизаторы имеют мультипроцессорную архитектуру: несколько мощных центральных процессоров выполняют функции вычисления таблицы маршрутизации, а менее мощные процессоры в интерфейсных модулях занимаются передачей пакетов на подключенные к ним сети и пересылкой пакетов на основании части таблицы маршрутизации, кэшированной в локальной памяти интерфейсного модуля.

3.5. Корпоративные модульные концентраторы.

Модульные корпоративные концентраторы представляют собой многофункциональные устройства, которые могут включать несколько десятков модулей различного назначения: повторителей разных технологий, коммутаторов, удаленных мостов, маршрутизаторов и т. п., которые объединены в одном устройстве с модулями-агентами протокола SNMP, и, следовательно, позволяют централизованно объединять, управлять и обслуживать большое количество устройств и сегментов, что очень удобно в сетях большого размера. Основная идея разработчиков таких устройств заключается в создании программно настраиваемой конфигурации сети. Модульный концентратор масштаба предприятия обычно обладает внутренней шиной или набором шин очень высокой производительности — до нескольких десятков гигабит в секунду, что позволяет реализовать одновременные соединения между модулями с высокой скоростью, гораздо большей, чем скорость внешних интерфейсов модулей (см. рис.). Ввиду того, что отказ корпоративного модульного концентратора приводит к очень тяжелым последствиям, в их конструкцию вносится большое количество средств обеспечения отказоустойчивости.

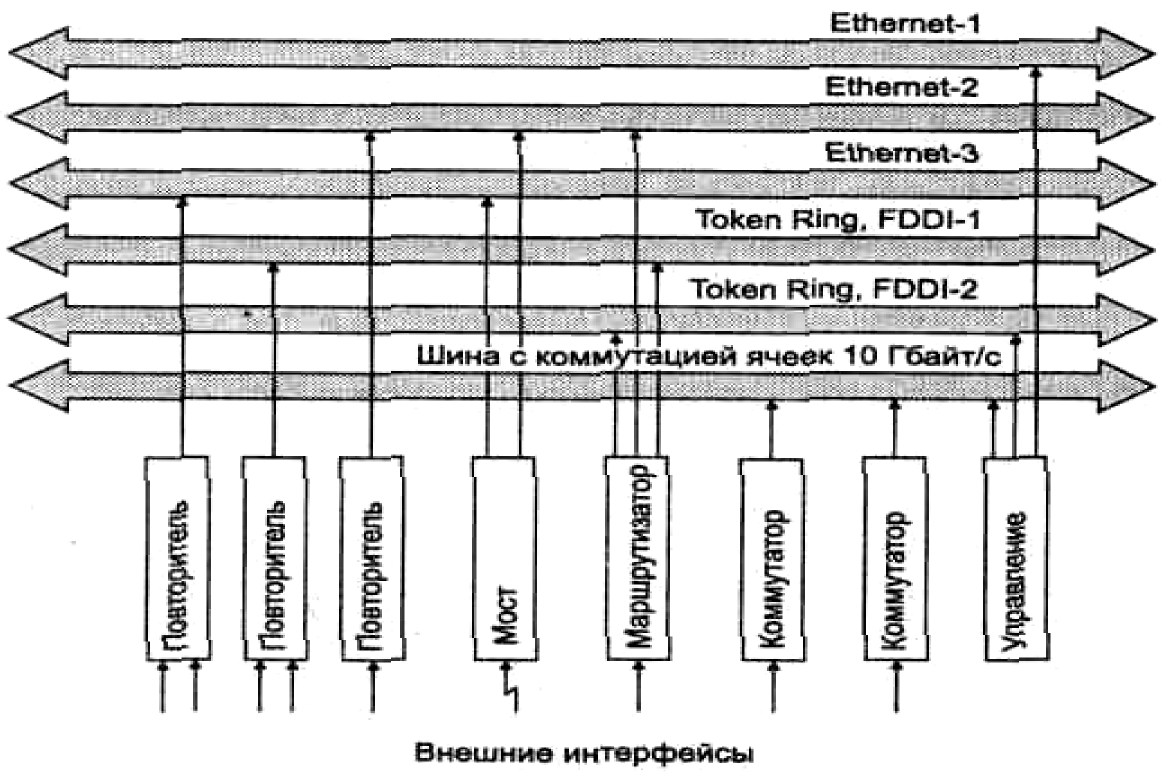


Рис. Структура корпоративного модульного концентратора

Примерами корпоративных концентраторов могут служить устройства System 5000 компании Nortel Networks, MMAC-Plus компании Cabletron Systems, CoreBuilder 6012 компании 3Com.

3.6. Коммутаторы третьего уровня.

В последние время наметилась тенденция сглаживания различий между коммутаторами и маршрутизаторами. Производители коммутаторов наделяют некоторые свои модели функциями маршрутизации, что позволяет использовать скорость коммутаторов и преимущества маршрутизаторов. Такие коммутаторы получили название коммутаторов третьего уровня. Функции коммутации и маршрутизации могут быть совмещены двумя способами:

1) Классическим, когда маршрутизация выполняется по каждому пакету, требующему передачи из сети в сеть, а коммутация выполняется для пакетов, принадлежащих одной сети.

2) Методом маршрутизации потока, когда маршрутизируется несколько первых пакетов устойчивого потока, а все остальные пакеты этого потока коммутируются.

Коммутаторы 3-го уровня с классической маршрутизацией.

Обычный коммутатор "прозрачен" для компьютеров сети, не имеет собственных MAC-адресов портов и захватывает все кадры, приходящие на порт, независимо от их адреса назначения, для последующей коммутации. Классический коммутатор 3-го уровня, подобно обычному коммутатору, захватывает все кадры своими портами независимо от их MAC-адресов, однако порты коммутатора 3-го уровня имеют и собственные MAC-адреса. Если захваченный кадр направлен на MAC-адрес какого-либо компьютера в сети, то пакет коммутируется. Если захваченный кадр направлен на MAC-адрес порта коммутатора, то пакет маршрутизируется. Коммутатор 3-го уровня может поддерживать динамические протоколы маршрутизации, такие как RIP или OSPF, а может полагаться на статическое задание маршрутов или на получение таблицы маршрутизации от другого маршрутизатора. Такие комбинированные устройства появились сразу после разработки коммутаторов, поддерживающих виртуальные локальные сети (VLAN). Для связи VLAN требовался маршрутизатор. Размещение маршрутизатора в одном корпусе с коммутатором позволяло получить некоторый выигрыш в производительности. Примерами таких коммутаторов могут служить хорошо известные коммутаторы LANplex (теперь CoreBuilder) 6000 и 2500 компании 3Com.

Коммутаторы 3-го уровня с маршрутизацией потоков.

Еще один тип коммутаторов 3-го уровня — это коммутаторы, которые ускоряют процесс маршрутизации за счет выявления устойчивых потоков в сети и обработки по схеме маршрутизации только нескольких первых пакетов потока. Последующие пакеты обрабатываются по схеме коммутации. Многие фирмы разработали подобные схемы, однако до сих пор они являются нестандартными, хотя работа над стандартизацией этого подхода идет в рамках одной из рабочих групп IETF.

Поток – это последовательность пакетов, имеющих некоторые общие свойства. По меньшей мере у них должны совпадать адрес отправителя и адрес получателя, и тогда их можно отправлять по одному и тому же маршруту. Желательно, чтобы пакеты потока имели одно и то же требование к качеству обслуживания (QoS, Quality of Service), т.е. одинаковые требования к скорости передачи данных, задержках в передаче пакетов, доле потерь пакетов и т.п. Приведем пример использования потоков для ускорения

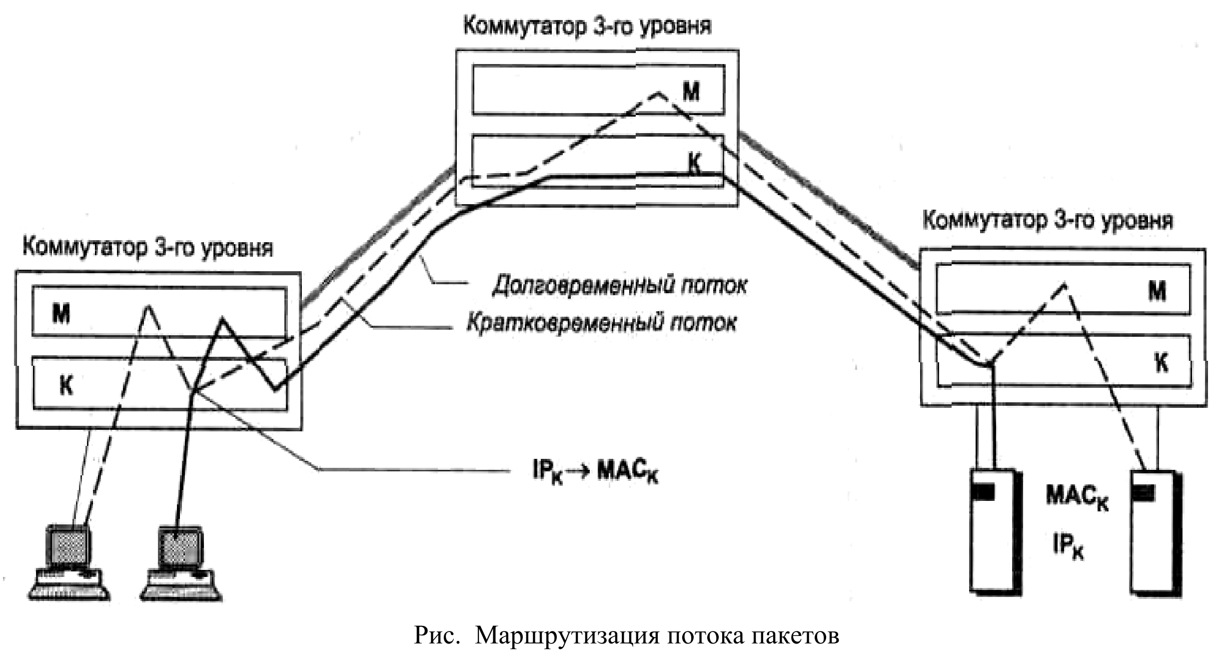


Рис. Маршрутизация потока пакетов

Если бы все коммутаторы 3-го уровня, изображенные на рис. , работали по классической схеме, то каждый пакет, отправляемый из рабочей станции, принадлежащей одной IP-сети, серверу, принадлежащему другой IP-сети, проходил бы через блоки маршрутизации всех трех коммутаторов. В схеме ускоренной маршрутизации такую обработку проходит только несколько первых пакетов долговременного потока, то есть классическая схема работает до тех пор, пока долговременный поток не будет выявлен. После этого первый коммутатор на пути следования потока выполняет нестандартную обработку пакета — он помещает в кадр канального протокола (например Ethernet) не MAC-адрес порта следующего маршрутизатора, а MAC-адрес узла назначения, который на рисунке обозначен как MAC_K. Как только эта замена произведена, кадр с таким MAC-адресом перестает поступать на блоки маршрутизации второго и третьего коммутатора, а проходит только через блоки коммутации этих устройств. Процесс передачи пакетов действительно ускоряется, так как они не проходят многократно повторяющиеся этапы маршрутизации. В то же время защитные свойства маршрутизаторы сохраняют, так как первые пакеты проверяются на допустимость передачи в сеть назначения, поэтому сохраняются фильтрация широковещательного шторма, защита от несанкционированного доступа и другие преимущества сети, разделенной на подсети.

Для реализации описанной схемы нужно решить несколько проблем. Первая – на основании каких признаков определяется долговременный поток. Это достаточно легкая проблема, и основные подходы к ее решению очевидны – совпадение адресов и портов соединения, общие признаки качества обслуживания, некоторый порог одинаковых пакетов для фиксации долговременности. Вторая проблема более серьезная. На основании какой информации первый коммутатор узнает MAC-адрес узла назначения? Этот узел непосредственно не подключен к сети первого коммутатора, поэтому использование протокола ARP здесь не поможет. Именно здесь расходятся пути большинства фирменных технологий ускоренной маршрутизации. Многие компании разработали собственные служебные протоколы, с помощью которых коммутаторы запрашивают этот MAC-адрес друг у друга, пока последний на пути коммутатор не выяснит его в своей сети, с помощью протокола ARP. Фирменные протоколы используют как распределенный подход, когда все коммутаторы равны в решении проблемы нахождения MAC-адреса, так и централизованный, когда в сети существует выделенный коммутатор, который помогает ее решить для всех.

Примерами коммутаторов 3-го уровня, работающими по схеме маршрутизации потоков, являются коммутаторы SmartSwitch компании Cabletron, а также коммутатор Catalyst 5000 компании Cisco, выполняющий свои функции совместно с маршрутизаторами Cisco 7500 по технологии Cisco NetFlow.

3.7. Шлюз (gateway), межсетевой экран (firewall), прокси-сервер, NAT.

Термин "шлюз" и термин "маршрутизатор" во многом схожи, но шлюз является более общим термином (всякий маршрутизатор является шлюзом). Шлюзом называется любое сетевое устройство, которое одновременно подключено к нескольким сетям при помощи нескольких сетевых интерфейсов, имеет в каждой сети свой адрес сетевого уровня и занимается продвижением пакетов между этими сетями. Например, шлюзом является компьютер одна сетевая карта которого подключена к сети 192.168.28.10.0 и имеет там IP-адрес 192.168.28.10.1, а другая сетевая карта подключена к сети 172.16.0.0 и имеет там IP-адрес 172.16.1.1. Шлюзом также будет являться и маршрутизатор. Даже обычный домашний компьютер, имеющий сетевую карту и модем можно рассматривать как шлюз, т.к. он имеет два интерфейса: один интерфейс – это интерфейс сетевой карты (локальной сети), IP-адрес которого может быть произвольным, а второй интерфейс – это интерфейс удаленного доступа (Internet), IP-адрес которого определяется провайдером, при подключении к нему по протоколу PPP.

Шлюз выполняет функции маршрутизации и продвижения пакетов между интерфейсами. Шлюзы также позволяют объединять разнородные (гетерогенные) сети, преобразуя, например, кадры Ethernet в кадры FDDI. Шлюз также является средством обеспечения безопасности подсети. Если сегмент сети соединен с остальной сетью через шлюз, то на шлюзе может быть установлен межсетевой экран (firewall, брандмауэр) – специальное программное обеспечение, которое контролирует как пакеты выходящие из данного сегмента, так и пакеты поступающие в данный сегмент. Межсетевой экран с фильтрацией пакетов уже был рассмотрен ранее (см. маршрутизаторы). Путем написания специальных правил, можно ограничить разрешенное взаимодействие между компьютерам сети и компьютерами сегмента. Правила имеют вид: "через шлюз допускается прохождение пакетов с IP-адресом отправителя 172.18.10.1 (порт 80) и IP-адресом получателя 192.168.1.1 (порт 21), в четверг с 15.00 до 19.00". Пакеты, не удовлетворяющие правилам фильтрации отбрасываются, а факт их наличия регистрируется в специальном журнале. Поскольку сервисы в сети связаны с определенными номерами портов, то закрыв входящие соединения на 23 порт, можно запретить извне управлять компьютерами сегмента по протоколу Telnet, а закрыв исходящие соединения на 80 порт можно запретить сотрудникам отдела (сегмента) просматривать Web-страницы.

Межсетевые экраны с фильтрацией пакетов просты, и в ряде случаев входят в состав самой операционной системы (например IPChains в ОС Linux). Однако межсетевые экраны с фильтрацией пакетов имеют и ряд недостатков:

- возможно задавать правила фильтрации по IP-адресам компьютеров, но не по имени пользователя.
- подсеть "видна" (маршрутизируется) извне.
- при выходе из строя межсетевого экрана подсеть становится незащищенной.

Для преодоления этих недостатков, в качестве межсетевого экрана используют прокси-сервера (proxy server). Прокси-сервер - это сервер посредник. Одно из назначений прокси-сервера - это ускорение работы сети, при подключении ее к Internet. Так например, кэширующий прокси-сервер Squid в ОС Linux сохраняет в достаточно большом кэше на диске Web-страницы, просмотренные разными пользователями, так что если какой-либо пользователь обратится к просмотренной кем-либо ранее странице, то эта страница не будет заново загружаться через Internet, а будет взята из кэша Squid. Прокси-сервер может использоваться и для сокрытия личности пользователя, путешествующего по Internet. Для этого пользователь сначала соединяется с анонимным прокси-сервером (например в Новой Зеландии), который получает пакеты от пользователя и перенаправляет их дальше от своего имени.

Установка прокси-сервера на шлюзе позволяет скрыть структуру подсети от внешней сети и реализовать гибкий межсетевой экран. При запрете продвижения IP-пакетов между интерфейсами шлюза, вся подсеть, с точки зрения внешней сети, представлена только одним IP-адресом – адресом прокси-сервера. Пользователь внешней сети, который хочет соединиться с компьютером внутри подсети, должен пройти следующую процедуру:

- Установить соединение с определенным портом прокси-сервера и указать имя компьютера внутри подсети с которым необходимо соединиться. Для каждого вида сервиса (http, ftp, smtp и т.д.) должна существовать своя программа-посредник, "прослушивающая" свой порт. Может существовать и универсальная программа - посредник, обслуживающая несколько сервисов. Если для какого-то сервиса программы-посредника нет (или она вышла из строя), то данный сервис будет не доступен. При установлении соединения возможно, хотя и не обязательно, проведение аутентификации (подтверждения личности) пользователя по его имени, паролю, IP-адресу. Возможна также регистрация факта и времени подключения в специальном журнале.
- Прокси-сервер создает соединение с компьютером внутри подсети, а затем обеспечивает обмен пакетами между внешней сетью и подсетью, подменяя адреса в проходящих через него пакетах. Возможно протоколирование соединения и фильтрация передаваемых данных (поскольку программа-посредник "понимает" протокол прикладного уровня своего сервиса).

Аналогичным образом происходит и соединение подсеть – внешняя сеть.

Другой технологией, позволяющей скрыть сеть предприятия, являеется NAT (Network Address Translation – трансляция сетевых адресов). NAT также позволяет компьютерам локальной сети работать с Internet через один IP-адрес. Технология осуществляет подмену IP-адресов отправителя и получателя, в проходящих через шлюз пакетах.

Поясним принцип работы NAT на примере. Допустим, имеется локальная сеть из десяти компьютеров. Все компьютера в сети имеют "серые" адреса 192.168.1.1 – 192.168.1.20, которые изолированы от сети Internet (не передаются маршрутизаторами Internet) и их не надо согласовывать с InterNIC. На компьютере "A", с IP-адресом 192.168.1.1 имеется модем, сетевой интерфейс которого, при подключении к Internet по протоколу PPP, автоматически получает от провайдера IP-адрес w1.x1.y1.z1. (запись условная). Таким образом, компьютер "A" имеет два сетевых интерфейса с адресами 192.168.1.1 и w1.x1.y1.z1. и является шлюзом локальная сеть – Internet.

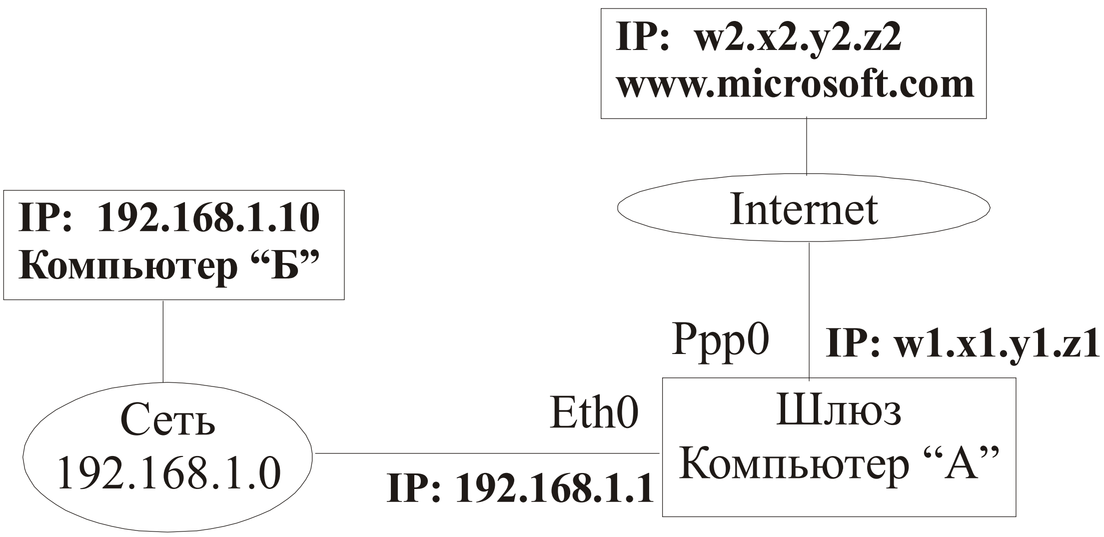


Рис. Технология NAT.

Пусть пользователь компьютера "Б" локальной сети (IP-адрес 192.168.1.10) обращается с помощью Web-браузера Internet Explorer к серверу www.microsoft.com, с IP-адресом w2.x2.y2.z2. (запись условная). Поскольку IP-адрес w2.x2.y2.z2. не относится к локальной сети, то на шлюз (компьютер "A") будет направлен IP-пакет со следующими данными:

- IP-адрес приемника: w2.x2.y2.z2 (www.microsoft.com)
- IP-адрес источника: 192.168.1.10 (компьютер "Б")
- Порт приемника: TCP-порт 80 (порт сервера Microsoft, протокол http)
- Порт источника: TCP-порт 1025 (порт компьютера "Б")

Этот IP-пакет перенаправляется протоколу NAT, который преобразовывает адреса исходящего пакета следующим образом.

- IP-адрес приемника: w2.x2.y2.z2 (www.microsoft.com)
- IP-адрес источника: w1.x1.y1.z1 (компьютер "А")
- Порт приемника: TCP-порт 80 (порт сервера Microsoft, протокол http)
- Порт источника: TCP-порт 5000 (порт компьютера "А")

При этом протокол NAT сохраняет в своей таблице преобразованных адресов запись:
"Адрес {192.168.1.10, порт 1025} заменен на адрес {w1.x1.y1.z1, порт 5000}".
Преобразованный IP-пакет отправляется по Интернету. Пакет, посылаемый в ответ на этот пакет, принимается протоколом NAT. Полученный пакет содержит следующие адресные данные.

- IP-адрес приемника: w1.x1.y1.z1 (компьютер "А")
- IP-адрес источника: w2.x2.y2.z2 (www.microsoft.com)
- Порт приемника: TCP-порт 5000 (порт компьютера "А")
- Порт источника: TCP-порт 80 (порт сервера Microsoft, протокол http)

Протокол NAT проверяет свою таблицу преобразованных адресов, после чего делает обратную замену:

- IP-адрес приемника: 192.168.1.10 (компьютер "Б")
- IP-адрес источника: w2.x2.y2.z2 (www.microsoft.com)
- Порт приемника: TCP-порт 1025 (порт компьютера "Б")
- Порт источника: TCP-порт 80 (порт сервера Microsoft, протокол http)

Преобразованный пакет направляется в локальную сеть.
Таким образом, протокол NAT подменяет адрес в IP-пакете, передавая его от своего имени, и пользуясь номером порта для того, чтобы "запомнить" какому компьютеру надо будет вернуть ответ на этот пакет. Шлюз не может одновременно создать с одного и того же своего порта два соединения с 80 портом сервера www.microsoft.com, т.к. для создания сокета необходимо, чтобы хотя бы один из параметров " IP-адрес отправителя, номер порта отправителя" – "IP-адрес получателя, номер порта получателя" у двух сетевых соединений не совпадали. Поэтому, если к серверу www.microsoft.com одновременно обратятся два компьютера локальной сети, то пакеты одного из них будут отправлены с порта 5000 (как в примере), а пакеты другого – будут отправлены, например, с порта 5001. При получении ответа от сервера www.microsoft.com, пакеты, поступившие на порт 5000, будут переправлены первому компьютеру, а пакеты, поступившие на порт 5001 – второму компьютеру.

При использовании NAT возникает следующая проблема: пакеты, содержащие IP-адрес только в заголовке пакета, правильно преобразовываются протоколом NAT, однако пакеты, содержащие IP-адрес в поле данных, могут неправильно преобразовываться при помощи NAT. Например, протокол FTP хранит IP-адрес в заголовке FTP для команды FTP PORT. Заголовок FTP, как и любой протокол прикладного уровня, хранится в поле данных IP-пакета. Если NAT не сможет правильно преобразовать IP-адрес из заголовка FTP и откорректировать поле данных, то могут возникнуть неполадки связи. Для устранения этой проблемы существуют редакторы NAT, работающие с заголовками прикладных протоколов. Не для всех протоколов необходим редактор NAT, например для протокола HTTP он не нужен. В ОС Windows 2000 реализованы редакторы NAT для следующих протоколов: FTP, ICMP, PPTP, NetBIOS через TCP/IP, RPC, Direct Play, H.323, Регистрация ILS на основе LDAP. Однако для зашифрованного трафика использование редактора NAT не предусмотрено, что следует учитывать при использовании NAT.

Лекция 4. Расчет корректности конфигурации локальной сети.

В данной лекции разговор пойдет о сетях Ethernet и Fast Ethernet, как наиболее распространенных, однако некоторые из рассмотренных ограничений будут справедливы и в других сетях (естественно, с другими значениями параметров).

4.1. Расчет корректности конфигурации сети Ethernet.

Для того, чтобы сеть Ethernet могла функционировать корректно, ее конфигурация должна удовлетворять определенным требованиям, которые включают в себя:

1) Ограничение на максимальную/минимальную длину кабеля.

Основным недостатком любого типа кабеля является затухание сигнала в кабеле. Если не использовать повторители (концентраторы), ретранслирующие и усиливающие сигнал, то расстояние между любыми двумя компьютерами в сети с топологией "общая шина" не может превышать некоторого предельного значения (см. табл.). При топологии "звезда" или "кольцо" это же ограничение накладывается на длину кабеля компьютер-компьютер или компьютер-концентратор (hub). Существуют также ограничения на минимальную длину кабеля между двумя сетевыми устройствами, что связано с физическими особенностями распространения сигнала в кабеле.

2) Ограничение на количество компьютеров в одном сегменте сети.

Сегмент образуют компьютеры, соединенными между собой при помощи повторителей (концентраторов). Два различных сегмента объединяются между собой при помощи мостов, коммутаторов, маршрутизаторов. В стандарте Ethernet предусмотрено ограничение на максимальное число компьютеров в одном сегменте сети (см. табл.).

3) Ограничение на число повторителей между любыми двумя компьютерами сети.

Число повторителей (концентраторов) между любыми двумя компьютерами в сети Ethernet не может быть больше четырех. Это ограничение называют "правилом четырех хабов". Ограничение связано с задержками в распространении сигнала, которые вносит повторитель (подробнее, объяснение см. в расчете PDV).

Таблица

Ограничения на конфигурацию сети Ethernet.

Стандарт	10Base-5	10Base-2	10Base-T	10Base-F
Кабель	Толстый коаксиальный кабель RG-8 или RG-11	Тонкий коаксиальный кабель RG-58	Неэкранированная витая пара категорий 3, 4, 5	Многомодовый волоконно-оптический кабель
Максимальная длина кабеля, м	500	185	100	2000
Минимальная длина кабеля, м	2,5	2,5	2,5	
Максимальное количество компьютеров в одном сегменте	100	30	1024	1024
Максимальное число повторителей между любыми станциями сети	4			
PDV не более	575 битовых интервалов			
PVV не более	49 битовых интервалов			

4) Ограничение на время двойного оборота сигнала (Path Delay Value, PDV).

Правило "четырех хабов" является достаточно простым, однако гарантирует корректность конфигурации сети с излишним "запасом". В некоторых случаях можно построить сеть и с большим числом повторителей между любыми двумя компьютерами в сети. Кроме того, правило четырех хабов не рассчитано на смешанные сети (коаксиал+витая пара+оптоволокно). Для более точной проверки используется расчет времени двойного оборота сигнала (PDV). Поясним термин PDV.

Четкое распознавание коллизий всеми компьютерами сети является необходимым условием корректной работы сети Ethernet. Если какой-либо передающий компьютер не распознает коллизию и решит, что кадр данных передан верно, то этот кадр данных будет утерян. Скорее всего, утерянный кадр будет повторно передан каким-либо протоколом верхнего уровня (транспортным или прикладным), но произойдет это через значительно более длительный интервал времени (иногда даже через несколько секунд), по сравнению с микросекундными интервалами, которыми оперирует протокол Ethernet. Поэтому если коллизии не будут надежно распознаваться узлами сети Ethernet, то это приведет к заметному снижению полезной пропускной способности сети.

Для надежного распознавания коллизий необходимо, чтобы передающий компьютер успевал обнаружить коллизию еще до того, как он закончит передачу этого кадра. Для этого время передачи кадра минимальной длины должно быть больше или равно времени, за которое сигнал коллизии успевает распространиться до самого дальнего компьютера в сети. Так как в худшем случае сигнал должен пройти дважды между наиболее удаленными друг от друга компьютерами в сети (в одну сторону проходит неискаженный сигнал, а на обратном пути распространяется уже искаженный коллизией сигнал), то это время называется временем двойного оборота (Path Delay Value, PDV).

Так как скорость распространения электрического сигнала конечна, то каждый метр кабеля вносит задержку в распространение сигнала. Существенную задержку также вносят повторители, вынужденные побитно принимать и усиливать сигнал. Для упрощения расчетов существует специальная таблица, содержащая величины задержек, указанных в битовых интервалах (см. табл.). Суммарная величина PDV, рассчитанная по таблице, не должна превышать 575 битовых интервалов. Для увеличения надежности сети, на случай отклонения параметров кабеля и повторителей, лучше оставлять "про запас" 4 битовых интервала, т.е. PDV не должно превышать 571 битовый интервал.

Таблица

Данные для расчета значения PDV

Тип сегмента*		Повторитель** левого сегмента, bt	Повторители промежуточного сегмента, bt	Повторитель правого сегмента, bt	Задержка*** среды на 1 м кабеля, bt	Максимальная длина сегмента, м
10Base-5		11,8	46,5	169,5	0,0866	500
10Base-2		11,8	46,5	169,5	0,1026	185
10Base-T		15,3	42,0	165,0	0,113	100
10Base-F	10Base-FB	—	24,0	—	0,1	2000
	10Base-FL	12,3	33,5	156,5	0,1	2000
	FOIRL	7,8	29,0	152,0	0,1	1000
AUI (> 2 м)		0	0	0	0,1026	2+48

* 10Base-FB, 10Base-FL, FOIRL – представляют собой различные варианты стандарта 10Base-F.

** В стандарте используются более точные термины: база левого, промежуточного и правого сегментов.

*** Для того, чтобы не нужно было два раза складывать задержки, вносимые кабелем, в таблице даются удвоенные величины задержек для каждого типа кабеля.

В таблице используются также такие понятия, как левый сегмент, правый сегмент и промежуточный сегмент. Левым сегментом считается сегмент компьютера-отправителя, а правым сегментом – сегмент компьютера-получателя (см. рис. 4.1.).

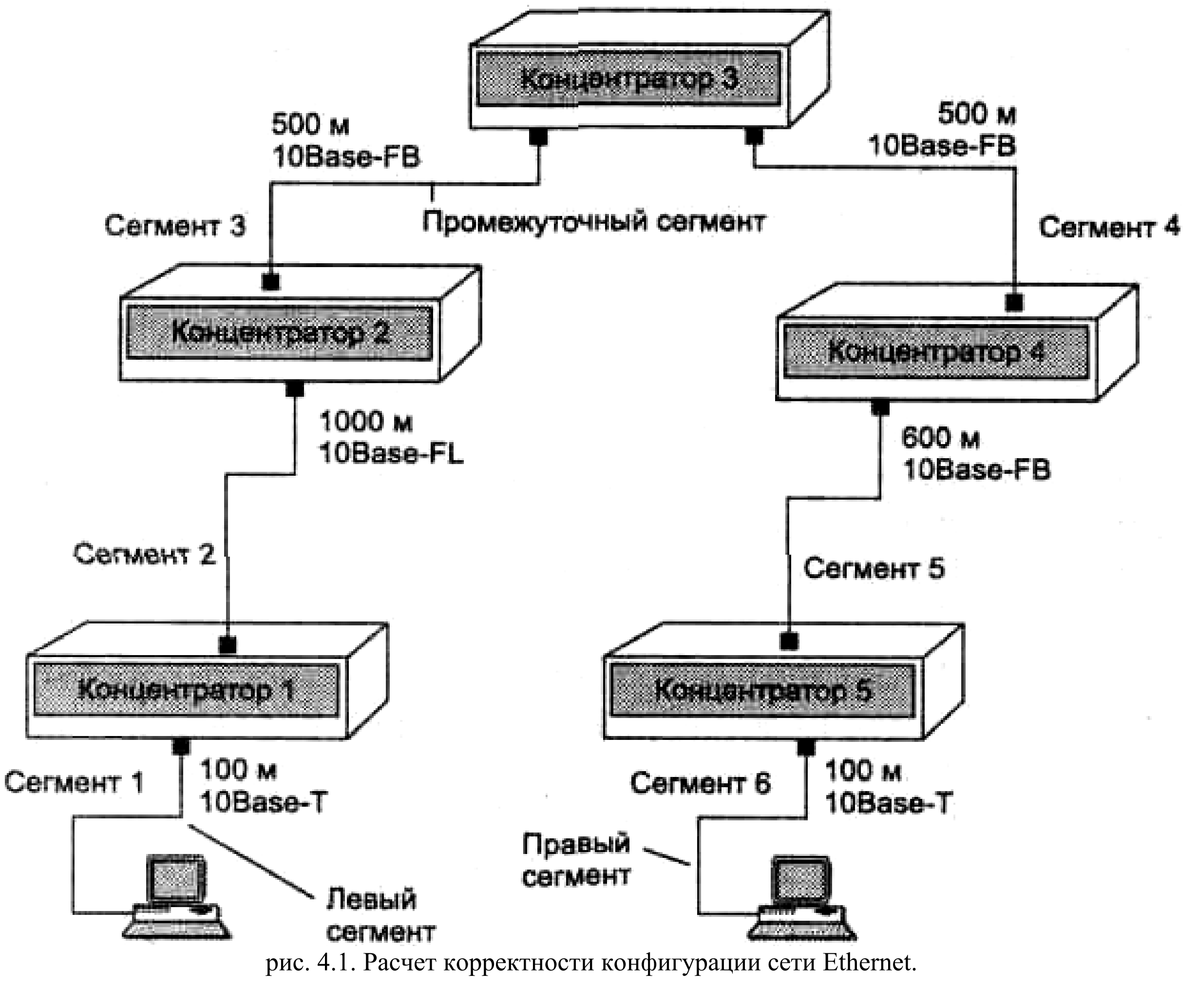


рис. 4.1. Расчет корректности конфигурации сети Ethernet.

Так как левый и правый сегменты имеют различные величины задержки повторителей, то в случае различных типов (коаксиал, витая пара, оптоволокно) сегментов на удаленных краях сети необходимо выполнить

расчеты дважды: один раз принять в качестве левого сегмента сегмент одного типа, а во второй — сегмент другого типа. Результатом считается максимальное значение PDV. В сети, изображенной на рисунке оба крайних сегмента принадлежат к одному типу (10Base-T, витая пара) поэтому двойной расчет не требуется. Расчет PDV для этой сети приведен ниже:
Левый сегмент 1: 15,3 (повторитель) + 100 x 0,113 (кабель) = 26,6.
Промежуточный сегмент 2: 33,5 (повторитель) + 1000 x 0,1 (кабель) = 133,5.
Промежуточный сегмент 3: 24 (повторитель) + 500 x 0,1 (кабель) = 74,0.
Промежуточный сегмент 4: 24 (повторитель) + 500 x 0,1 (кабель) = 74,0.
Промежуточный сегмент 5: 24 (повторитель) + 600 x 0,1 (кабель) = 84,0.
Правый сегмент 6: 165 (повторитель) +100 x 0,113 (кабель) = 176,3.
Итого PDV: 568,4 битовых интервала.
Так как расчетное значение PDV меньше максимально допустимой величины 575, то эта сеть проходит по критерию времени двойного оборота сигнала несмотря на то, что количество повторителей — больше 4-х. Однако, чтобы признать конфигурацию сети корректной, нужно также рассчитать уменьшение межкадрового интервала (PVV).

5) Ограничение на сокращение межкадрового интервала (Path Variability Value, PVV).
При отправке кадра, компьютеры обеспечивают начальное межкадровое расстояние в 96 битовых интервала. При прохождении через повторители, межкадровый интервал уменьшается. Суммарное сокращение межкадрового интервала (PVV) не должно превышать 49 битовых интервалов. Для расчета PVV также существует таблица.

Таблица

Сокращение межкадрового интервала повторителями

Тип сегмента	Передающий сегмент, bt	Промежуточный сегмент, bt
10Base-5 или 10Base-2	16	11
10Base-FB	—	2
10Base-FL	10,5	8
10Base-T	10,5	8

В соответствии с данными таблицы, рассчитаем значение PVV для нашего примера.
Левый сегмент 1 10Base-T: сокращение в 10,5 bt.
Промежуточный сегмент 2 10Base-FL: 8.
Промежуточный сегмент 3 10Base-FB: 2.
Промежуточный сегмент 4 10Base-FB: 2.
Промежуточный сегмент 5 10Base-FB: 2.
Итого PVV: 24,5 битовых интервала.
Рассчитанное значение PVV 24,5 меньше предельного значения в 49 битовых интервала. В результате, приведенная в примере сеть соответствует стандартам Ethernet по всем параметрам, хотя и включает в себя более четырех повторителей.

4.2. Расчет корректности конфигурации сети Fast Ethernet.

Порядок расчета корректности конфигурации сети Fast Ethernet несколько отличается от расчета сети Ethernet, как по параметрам, так и по схеме расчета. Стандарт Fast Ethernet не поддерживает коаксиальный кабель и сеть строится исключительно по топологии звезда. Ограничения на длину кабеля компьютер-повторитель, компьютер-компьютер приведены ниже:

Таблица

Ограничения на длину кабеля в стандарте Fast Ethernet.

Тип кабеля	Стандарт	К повторителю подключен	Максимальная длина кабеля, м
Витая пара категории 5	100Base-TX	–	100 м
Витая пара категории 3, 4	100Base-T4	–	100 м
Многомодовое оптоволокно 62,5/125 мкм	100Base-FX	только оптоволоконный кабель	412 м (полудуплекс) 2000 м (полный дуплекс)
		один оптоволоконный кабель и несколько кабелей витая пара	160 м
		несколько оптоволоконных кабелей и несколько кабелей витая пара	136 м

Ограничение на количество повторителей
Повторители Fast Ethernet делятся на два класса. Повторители класса 1 имеют порты всех типов (стандарт 100Base-TX, 100Base-FX и 100Base-T4). Повторители класса 2 имеют либо все порты 100Base-T4, либо порты 100Base-TX и 100Base-FX. Между любыми двумя компьютерами в сети может быть не более двух

повторителей класса 2 или только один повторитель класса 1. Между собой повторители класса 1 должны объединяться при помощи коммутаторов, мостов, маршрутизаторов. Пример конфигурации сети Fast

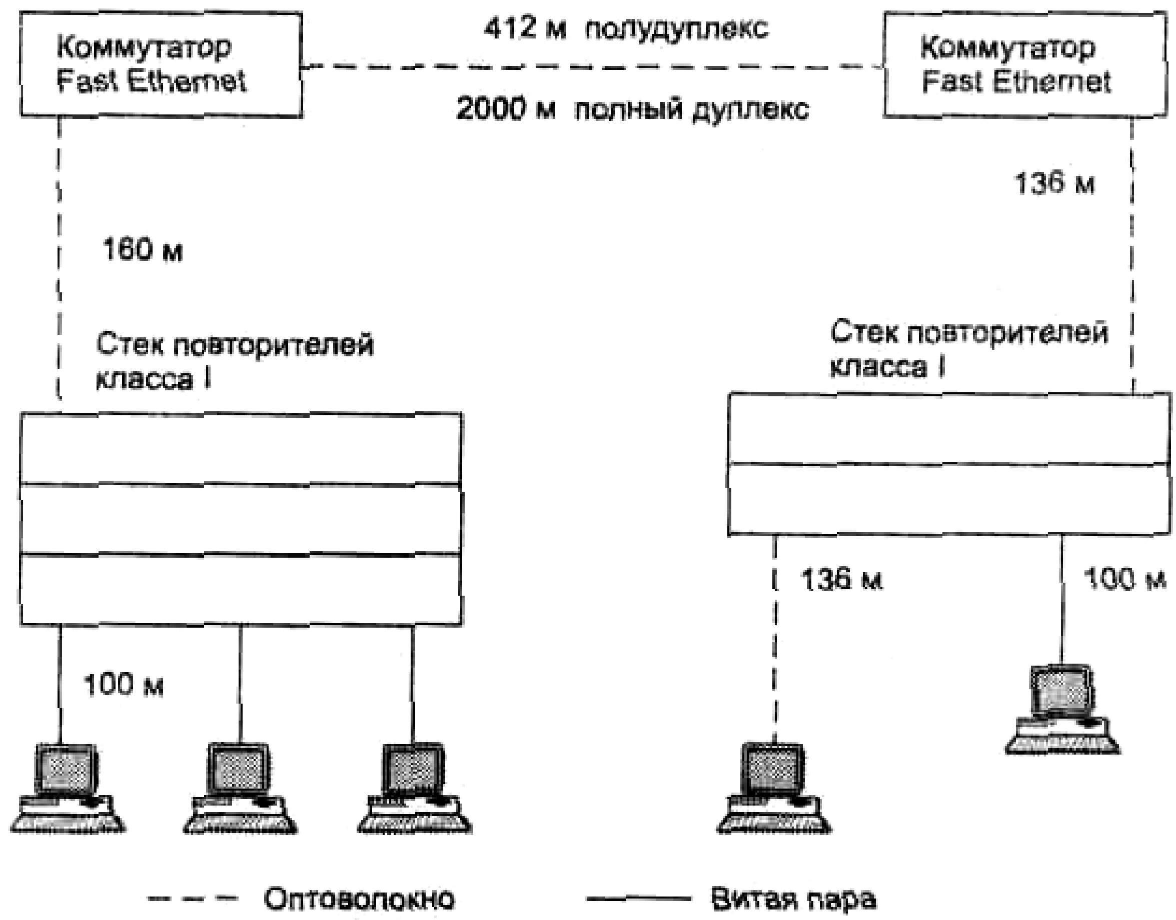


рис. 4.2. Пример конфигурации сети Fast Ethernet.

Приведенных правил построения сети вполне достаточно для определения корректности конфигурации сети, т.к. эти правила выбраны с минимальным "запасом прочности". Однако, при желании, можно провести и расчет PDV, исходя из следующих соображений. Максимально допустимая величина PDV = 512 битовых интервала. При расчете сегменты не делятся на правый и левый. Для расчета берутся задержки, которые вносят две взаимодействующих через повторитель сетевые карты компьютеров (или сетевая карта компьютера и порт коммутатора). Также учитывается задержка сигнала в повторителе и задержка, вносимая кабелем. Исходные данные для расчета приведены в таблице .

Таблица

Расчет задержек распространения сигнала.

Задержка, вносимая кабелем		Задержка, вносимая сетевыми картами		Задержка, вносимая повторителем	
Тип кабеля	Удвоенная задержка, bt на 1 м	Тип сетевых карт, взаимодействующих через повторитель	Удвоенная задержка, bt	Класс повторителя	Удвоенная задержка, bt
UTP Cat 3	1,14 bt	Два адаптера TX/FX	100 bt	1	140
UTP Cat 4	1,14 bt	Два адаптера T4	138 bt	2	92
UTP Cat 5	1,112 bt	Один адаптер TX/FX и один T4	127 bt		
Оптоволокно	1,0 bt				

Рассчитаем, для примера, PDV между двумя компьютерами, подключенными к повторителю 1 класса, расположенному в правой части рис. 4.2. Предполагаем, что используется витая пара 5-ой категории (TX).
PDV=100*1,112 (кабель) + 136*1,0 (кабель) + 100 (сетевые карты) + 140 (повторитель) = 487,2 < 512
Рассчитанное значение PDV меньше предельного значения в 512 битовых интервалов, соответственно расчеты пока не выявили некорректность конфигурации сети. Для признания конфигурации корректной, необходимо продолжить расчеты для остальных пар компьютер-компьютер, компьютер-коммутатор.

Лекция 5. Windows 2000

Windows 2000 (NT 5.0, или на слэнге w2k) - операционная система фирмы Microsoft, основанная на технологии Windows NT 4.0. Была создана группой разработчиков под руководством Дэйва Катлера, который еще в 1988 году пришёл в Microsoft (ранее работал в DEC) специально для работы над NT 4.0

Существуют следующие варианты Win2000:

1. Windows 2000 Professional - вариант для рабочей станции = Windows NT Workstation.
2. Windows 2000 Server или Windows 2000 Advanced Server - варианты для сервера = Windows NT Server. Наилучший выбор - Advanced Server: более полный пакет программ, поддерживает до 8 процессоров.
3. Windows 2000 Data Center – вариант для мощных корпоративных серверов (поддержка до 64 процессоров, в небольших сетях будет только зря использовать ресурсы).

5.1. Отличительные особенности Win2000

Windows 2000 – это мощная и удобная операционная система, которая имеет свои достоинства и недостатки. Сопоставление Windows 2000 и Unix/Linux систем, с точки зрения выбора операционной системы для сервера локальной сети, приведены ниже.

Таблица 5.1.

Сопоставление Windows 2000 и Unix/Linux систем

Windows 2000	Unix/Linux
Удобный и привычный для пользователей Windows графический интерфейс. Большинство настроек выполняется при помощи удобных графических форм (флажки, переключатели и т.п.).	В последнее время большинство *nix систем получило графический интерфейс, однако пока еще большое значение (и одновременно преимущество) имеет управление сервером из командной строки, и использование простых текстовых редакторов для ручного редактирования файлов конфигурации.
Большинство задач по конфигурированию и управлению сервером выполняется при помощи удобных мастеров. Для настройки системы необязательно иметь детальное представление о принципах ее работы. Сложно допустить фатальную ошибку.	Большинство действий по конфигурации системы выполняются вручную. Невозможно правильно настроить систему, не разобравшись перед этим во всех деталях того, что Вы собираетесь делать. Пользователь с низким уровнем подготовки может легко допустить ошибку, критическую для сетевой безопасности или стабильной работы системы.
Справочная система рассчитана на пользователя с низким и средним уровнем подготовки.	Справочная система рассчитана на пользователя со средним и высоким уровнем подготовки.
Хорошая локализация. Настройка русского, или других национальных языков, не представляет проблем.	Хотя даже между различными дистрибутивами Linux в этом вопросе имеются различия, однако в целом локализация реализована плохо. Настройка русского языка представляет собой не тривиальную проблему. Отсутствует единая схема настройки. Приходится настраивать русский язык в нескольких местах и даже по отдельности в разных программных пакетах.
Чрезвычайно требовательна к ресурсам оборудования (объем памяти, процессор, дисковое пространство и т.д.).	Мало требовательна к ресурсам оборудования. Если на том же компьютере, где установлена Windows 2000, установить Linux, то система будет работать в несколько раз быстрее.
Хотя Windows 2000 защищена существенно больше, чем Windows 95/98/ME, однако и она имеет существенные проблемы с сетевой безопасностью. В целом уязвима вся операционная система, хотя наиболее часто атакам подвергается Internet Information Services (IIS), предоставляющий доступ к сервисам HTTP, FTP, SMTP и др.	Гораздо меньше, чем Windows 2000 уязвима к сетевым атакам. Большинство из атак хорошо изучены и отработаны механизмы противодействия им. Если не пользоваться новым не стабильным и не протестированным программным обеспечением, а также грамотно конфигурировать сервер, то Вы будете достаточно надежно защищены от сетевых атак.
За использование лицензионной копии операционной системы необходимо платить достаточно большую сумму. Исходные тексты операционной системы не доступны и не могут быть изменены под нужды пользователя.	Операционная система и программное обеспечение для нее распространяется бесплатно и доступна в исходных текстах. Наличие исходных текстов и сама архитектура системы позволяет сконфигурировать ОС "под себя" оставив только то, что действительно необходимо (это значительно повышает и без того высокое быстродействие ОС). Открытые исходные тексты позволяют любому желающему тестировать и исправлять ошибки ОС. В результате, стабильность и надежность работы Unix/Linux значительно превосходят Windows 2000. Наличие исходных текстов ОС и программ позволяет проверить их на наличие "черных ходов" и других нарушений сетевой безопасности, не доверяя честному слову разработчиков.

Исходя из изложенного выше можно сделать вывод: область применения Windows 2000 – это локальные сети небольших предприятий, где персонал администрирования сети имеет низкую или среднюю квалификацию, либо не хочет тратить значительное время на настройку и обслуживание ОС, при условии, что обеспечение сетевой безопасности не является ключевым моментом политики предприятия, и возможные последствия от инцидентов с безопасностью не существенны. Если же сетевая безопасность важна и имеется высококвалифицированный персонал администрирования сети, то лучше использовать Unix/Linux системы. В частности, шлюз в Internet настоятельно рекомендуется реализовывать на базе Unix/Linux.

Из сказанного вовсе не следует, что Linux однозначно лучше Windows 2000. Просто каждая операционная система имеет свою область применения и свой набор возможностей, из которых необходимо выбрать то, что Вам необходимо. Ниже более детально будут рассмотрены некоторые возможности ОС Windows 2000 Server. Рассмотрению возможностей ОС Linux будет посвящена отдельная лекция.

5.2. Файловая система NTFS

NTFS выросла из файловой системы HPFS, разрабатываемой совместно IBM и Microsoft для проекта OS/2. Отличительными характерными чертами NTFS являются:

1. Надежность: вызвать сбой в NTFS чрезвычайно сложно (для опыта запускалась много различных приложений, оптимизаторы диска, и в самые неподходящие моменты жалась кнопка reset - повторение этого эксперимента добрый десяток раз никакого впечатления на систему не произвело). NTFS содержит две копии аналога FAT, которые называются MFT (Master File Table). Если оригинал MFT повреждён (например, при появлении bad-сектора), то система использует копию MFT. В отличие от FAT MSDOS, технология MFT больше напоминает обработку транзакций в базах данных: при любом сбое во время записи файла на диск, MFT будет восстановлена в состояние "до записи файла". Таким образом, вы теряете не весь файл, а только те изменения, которые находились в момент сбоя в памяти или в кэше контроллера, и не успели записаться на диск.
2. Защищенность: NTFS рассматривает файлы, как объекты. Каждый файловый объект обладает методами (например, open, close, read и write) и свойствами (имя, дата создания, дата последнего обновления, архивный статус, дескриптор безопасности). Дескриптор безопасности позволяет настроить права доступа к объекту и аудит объекта (Проводник/Выделить файл (каталог)/Контекстное меню/Свойства/Безопасность). Для различных пользователей (групп пользователей) возможно определить следующие права доступа: полный доступ, запись, чтение, обзор содержимого папки, создание подпапок и файлов, удаление подпапок и файлов, выполнение файлов, чтение и запись атрибутов и разрешений файла (папки), смена владельца и др. Там же, для каждого из пользователей (групп пользователей) можно создать политику аудита, которая будет регистрировать в специальном журнале (Пуск/Панель управления/Администрирование/Просмотр событий/Журнал безопасности) успех или неудачу при попытке получить доступ к файлу (каталогу). Попытки доступа классифицируются в соответствии с приведенным выше перечнем прав доступа. Регистрация успешных попыток позволяет следить за разрешенной деятельностью пользователей, регистрация неудачных попыток позволяет выявить попытки нарушения прав доступа. Для обеспечения большей надежности можно также использовать шифрованную файловую систему (Encrypted File System, EFS). Эта возможность встроена в Windows 2000 (Проводник/Выделить файл (каталог)/Контекстное меню/Свойства/Общие/Атрибуты/Другие/Шифровать содержимое для защиты данных). EFS использует шифрование открытого ключа (асимметричный алгоритм шифрования), что позволяет администратору создать агента восстановления зашифрованных данных – человека, который сможет расшифровать файлы других пользователей (например, при увольнении работника, по решению суда, при утере работником ключа шифрования из-за сбоев на диске).
3. Работа с большими дисками размером до 16,777,216 терабайт. Сжатие данных встроено на уровне файловой системы и позволяет сжимать не только целиком диск, но также отдельные каталоги и файлы "прозрачно" для пользователя (Проводник/Выделить файл (каталог)/Контекстное меню/Свойства/Общие/Атрибуты/Другие/Сжимать содержимое). Более высокая скорость работы с диском, благодаря структуре файловой системы и Microsoft Index Server, который значительно ускоряет поиск файлов, за счёт индексации содержимого дисков. Возможность поиска файла, по имени его владельца. Например, вам нужно удалить все файлы созданные уволенным сотрудником, а их на диске – тысячи.
4. Возможность "квотирования", т.е. ограничения максимального размера, выделяемого пользователю на диске, причем файлы пользователя могут находиться в самых разных каталогах, но их суммарный объем не может превышать установленного администратором (Проводник/Выделить диск/Свойства/Квота).
5. Монтирование сетевых и локальных дисков в любой каталог файловой системы (аналогично монтированию в Unix/Linux). Возможность изменить букву диска, или полностью удалить букву диска и оставить доступ к диску только через каталог на другом диске – таким образом можно организовать доступ к большому количеству дисков через единую структуру каталогов (Пуск/Настройка/Панель управления/Администрирование/Управление компьютером/Запоминающие устройства/Управление дисками/Выбрать диск/Контекстное меню/Изменение буквы диска и пути диска). Данная возможность доступна не только для дисков NTFS, но и для дисков с другими файловыми системами.

5.3. Распределенная файловая система DFS

Распределенная файловая система (DFS) дает возможность предоставить пользователям файлы, физически находящиеся на разных серверах, так, как если бы они находились в одном месте. Например, если бухгалтерская документация находится на разных серверах, можно использовать DFS, чтобы для пользователей все выглядело так, как будто вся документация располагается на одном сервере. Или, например, в рамках DFS, можно переместить файл с одного сервера на другой без необходимости информировать пользователей о том, что "файл переехал" – для них файл останется там же, где и был. Кроме того, задачи по обслуживанию сервера (обновлению программ и т.п.), могут выполняться без отключения пользователей. Настройка файловой системы DFS осуществляется через меню Пуск/Настройка/Панель управления/Администрирование/Распределенная файловая система DFS.

5.4. Динамические диски в Windows 2000

Win2000 позволяет преобразовать винчестер (физический диск) в динамический диск (Пуск/Настройка/ Панель управления/ Администрирование/Управление компьютером/Запоминающие устройства/Управление дисками/ Выбрать ***физический*** диск (например, диск 0)/Контекстное меню/Обновление до динамического диска). Внимание! Эта операция полностью передает физический диск в распоряжение Windows 2000 и ***удаляет*** с него логические разделы других операционных систем. Динамический диск не может содержать разделы или логические диски. Данные с динамического диска могут быть считаны только при помощи Windows 2000. Создание динамических дисков позволяет:

1. Создавать составные (spanned) динамические диски, т.е. несколько винчестеров, ведущих себя так, как если бы это был один большой винчестер. Данные пишутся последовательно, т.е. сначала заполняется один винчестер, затем второй и т.д.
2. Создавать чередующиеся (stripped) динамические диски, т.е. несколько винчестеров, ведущих себя как один большой винчестер, причем данные пишутся на диски параллельно, что ускоряет дисковые операции. Чисто условно можно представить себе так: 1-3-5 кластер файла пишется на первый диск, а 2-4-6 кластер – на второй диск.
3. Создавать зеркальные (mirrored) диски. Данные записываемые на один из дисков автоматически дублируются на другом. Это обеспечивает большую надёжность сохранности данных.
4. Создавать RAID диски. RAID – состоит из трёх, или более дисков. Данные пишутся параллельно на два или более дисков (см. п.2), а на третий диск записывается код коррекции ошибок (ECC) , с помощью которого можно восстановить содержимое испорченного блока на одном из дисков данных, по информации уцелевшего блоков второго диска данных. Или, чисто условно, зная содержимое кластеров 1-3-5 на первом диске и коды коррекции ошибок ECC1-ECC2-ECC3 с третьего диска, можно восстановить содержимое кластеров 2-4-6 второго диска. Эта технология экономнее, чем создание зеркальных дисков (дублируются не все кластеры 1-2-3-4-5-6, а только их половина ECC1-ECC2-ECC3), однако работает медленнее.

5.5. Служба каталогов Active Directory в Windows 2000 (ранее NTDS в Win NT 4.0), сценарии входа и профили пользователя.

Active Directory - это новое средство централизованного управления пользователями и сетевыми ресурсами, облегчающее администрирование больших сетей. Вся сеть представляется в виде иерархической структуры каталогов (контейнеров). Преимуществом является то, что все пользователи (группы пользователей, компьютеры, принтеры и т.д.) регистрируются не на каждом компьютере сети по отдельности, а централизованно – в службе каталогов Active Directory. После этого пользователь может подойти к любому компьютеру в офисе, ввести свой пароль, и перед ним будет его рабочий стол, его документы, его настройки. При использовании Active Directory у администратора отпадает необходимость вручную конфигурировать каждый компьютер, если, к примеру, необходимо поменять права доступа к какому-либо объекту сети или установить новый сетевой принтер. Такие изменения можно производить сразу для всей сети.

При использовании Active Directory вся информация о сети (а точнее о домене сети) хранится на специальном сервере – контролере домена. Контролеров домена (серверов) в одном домене может быть несколько, что повышает отказоустойчивость системы. При подключении к сети, пользователь общается именно с контролером домена, передавая ему свое имя и пароль (подробнее система безопасности Windows 2000 будет рассмотрена ниже). Создание службы Active Directory означает ее установку на контролер домена. Контролер домена должен работать под управлением минимум Windows 2000 Server. Рабочие станции под Windows 2000 Professional могут работать в среде Active Directory, но не могут создавать её. Создание Active Directory осуществляется при помощи команды меню "Пуск/Настройка/Панель управления/Администрирование/Настройка сервера/Active Directory". После создания Active Directory управление учетными записями пользователей доступно через команду меню "Пуск/Настройка/Панель управления/Администрирование/Active Directory – пользователи и компьютеры" (подробнее см. в разделе "Система безопасности Windows 2000"). При помощи этого меню можно создавать/удалять пользователей, менять их пароль, членство в различных группах и т.д.

Как уже упоминалось выше, при использовании Active Directory, пользователь может подойти к любому компьютеру в сети, ввести свое имя и пароль и Windows 2000 сам установит все настройки рабочего стола пользователя, подключит сетевые диски с документами пользователя и т.д. Достигается это за счет использования сценариев входа и профилей пользователя.

Профиль пользователя определяет настройки рабочей среды, включая настройки рабочего стола и меню пользователя, настройки дисплея, сеть, соединения с принтером, содержимое реестра и другие установки. Существуют следующие типы профилей пользователя:

- локальный профиль — создается при первом входе пользователя на конкретный компьютер и хранится на локальном жестком диске конкретного компьютера. Любые изменения локального профиля (настройка меню и т.п.) будут применены к данному компьютеру.
- перемещаемый профиль — создается системным администратором и хранится на сервере. При входе пользователя на любой компьютер в сети, он может использовать этот профиль и получить все стандартные настройки своего рабочего места, вне зависимости от того, с какого компьютера он вошел в сеть. Любые изменения перемещаемого профиля будут обновлены на сервере.
- обязательный профиль — является перемещаемым профилем, который не может быть изменен пользователем. Пользователь по-прежнему может настраивать свой рабочий стол и т.д., однако после выхода из системы эти изменения будут утеряны и следующий раз снова загрузится старый экземпляр профиля пользователя. Только системные администраторы могут вносить изменения в обязательный профиль.

Локальный профиль пользователя создается при первом сеансе работы пользователя за данным компьютером и хранится в папке "Documents and Settings\Имя пользователя". Перемещаемый профиль создается администратором. Создание перемещаемого профиля:

1) Создать на сервере папку (например, D:\Перемещаемые_профили) и через "Контекстное меню/Доступ" присвоить ей сетевое имя (например, Профили) и открыть к ней полный общий доступ для группы "пользователи домена" (кнопка "Разрешения").
2) Скопировать в эту папку локальный профиль пользователя с какого-либо компьютера в сети при помощи "Проводника", или меню "Пуск/Настройка/Панель управления/ Система/Двойной щелчок/ Профили пользователей/Выделить нужный профиль/Кнопка копировать/ Копировать профиль на" – указать сервер и каталог, куда будет скопирован профиль. Установить на скопированный каталог и подкаталоги разрешения "Полный доступ" только для данного пользователя и группы "администраторы" (Выделить каталог/Контекстное меню/Свойства/Безопасность).
3) В меню "Пуск/Настройка/Панель управления/Администрирование/Active Directory – пользователи и компьютеры/Выделить пользователя/Двойной щелчок/Профиль/Путь к профилю" указать путь к перемещаемому профилю. Необходимо указывать полный сетевой путь к файлам профиля (например, \\имя_сервера\Профили\имя_пользователя).

Обязательный профиль создается аналогично перемещаемому профилю за исключением того, что создается отдельный общий сетевой ресурс (например, Обязательные_профили), с доступом только "Чтение", а каталог с обязательным профилем должен носить расширение ".man" (например, \\имя_сервера\Обязательные_профили\имя_пользователя.man) и иметь для пользователя разрешения только "чтение и выполнение, список содержимого папки". Для администраторов сохраняются разрешения "полный доступ".

Примечание 1: Создать обязательный профиль можно и переименовав скрытый файл ntuser.dat, находящийся в основном каталоге профиля, в файл ntuser.man и установить на него разрешения "только чтение". Файл NTuser.dat отображает параметры реестра операционной системы Windows 2000.
Примечание 2: Операционная система Windows 2000 не поддерживает использование зашифрованных файлов совместно с перемещаемыми профилями. Файлы профиля не должны быть зашифрованы при помощи EFS.

Профили позволяют настроить параметры среды пользователя, однако не позволяют выполнить определенные ***<u>действия</u>*** (например, подключить сетевой диск). Для этих целей используют сценарии входа пользователя. Сценарий входа – это небольшая программа, которая запускается автоматически при входе пользователя на компьютер. Как правило, сценарий входа в систему представляет собой пакетный файл с расширением ".bat", однако допускается использование и любой исполняемой программы (расширение ".exe"), а также программ на языках JavaScript (расширение ".js") и программ на VBScript (расширение ".vbs"). Программы с расширениями js и vbs могут запускаться благодаря серверу сценариев Windows "Cscript.exe" (или "Wscript.exe").

В сценарии могут использоваться переменные среды. Ниже показаны примеры таких переменных, для использования в bat-файлах (в exe- js- и vbs- программах эти переменные также могут использоваться, однако синтаксис обращения к ним будет отличаться).

Таблица 5.2.

Некоторые переменные среды

Переменная*	Описание
%USERNAME%	Имя пользователя.
%USERPROFILE%	Профиль пользователя.
%HOMEPATH%	Полный путь к основному каталогу пользователя.
%USERDOMAIN%	Имя домена, содержащего учетную запись пользователя.
%HOMEDRIVE%	Имя диска на локальном компьютере пользователя, связанного с основным каталогом пользователя.
%OS%	Операционная система, используемая пользователем.
%SYSTEMROOT%	Корневой каталог Windows.
%COMSPEC%	Имя командного процессора.
%PATH%	Путь поиска выполняемых файлов.
%PATHEXT%	Расширения для поиска выполняемых файлов (com, exe).
%TEMP%	Каталог временных файлов.
%PROCESSOR_ARCHITECTURE%	Тип процессора рабочей станции пользователя (например 80386).
%PROCESSOR_LEVEL%	Уровень процессора рабочей станции пользователя. Например, для Pentium III это значение равно 6.
%PROCESSOR_IDENTIFIER%	Идентификатор процессора. Например x86 Family 6 Model 7 Stepping 3, GenuineIntel.

* Формат %имя_переменной% используется только в bat-файлах.

Пример сценария (файл scenario.bat):

```
@echo off
echo Доброе пожаловать %USERNAME% в домен %USERDOMAIN%
echo .
pause
```

Для создания сценария входа необходимо поместить файл сценария в каталог Scripts (обычно "C:\WINNT\SYSVOL\sysvol\имя_домена\scripts" (сетевое имя NETLOGON) или "C:\Winnt\System32\Repl\Import\Scripts", а затем в меню "Пуск/Настройка/Панель управления/Администрирование/Active Directory - пользователи и компьютеры/ Выделить пользователя/Двойной щелчок/Профиль/Сценарий входа" указать название файла сценария (например, scenario.bat). Если перед именем файла сценария указан относительный путь к файлу (например Admins\scenario.bat), то поиск файла проводится в указанном подкаталоге локального каталога сценариев.

5.6. Службы DNS, WINS, DHCP

Служба DNS отвечает за преобразование URL-адресов (типа www.microsoft.com) в IP-адреса. Сервер DNS интегрирован в Windows 2000, что позволяет использовать в локальной сети тот же формат имен компьютеров, принтеров и др. ресурсов, что и в Интернет. В результате исчезает разница между локальной сетью и Интернет. Например, набрав URL-адрес принтера, пользователь может обратиться к сетевому принтеру локальной сети также, как бы он обратился к любому ресурсу в Интернет. Настройка сервера DNS осуществляется посредством меню Пуск/Настройка/Панель управления/Админстрирование/DNS.

Служба WINS использует централизованную базу данных для установления соответствия между NetBIOS-именами и IP-адресами в сети (напомним, что "Сетевое окружение" в ОС Windows использует именно протокол NetBIOS). Настройка сервера осуществляется через меню Пуск/Настройка/ Панель управления/Администрирование/WINS.

Служба DHCP (Dynamic Host Configuration Protocol) используется для динамической настройки IP-адресов компьютеров сети. Каждый компьютер, работающий в сети на основе протокола TCP/IP, должен иметь уникальный IP-адрес. Если быть более точным, то IP-адрес получает не сам компьютер, а сетевые интерфейсы, которые установлены на компьютере. IP-адрес может быть статическим или динамическим. Статический IP-адрес назначается вручную, в меню Пуск/Настройка/Панель управления/Сеть и удаленный доступ к сети/Выбрать название сетевого подключения (сетевой интерфейс)/Контекстное меню/Свойства/ Общие/ Протокол Интернета TCP/IP /Свойства/ Использовать следующий IP-адрес. На этой же вкладке назначаются IP-адреса серверов DNS и WINS, маршрутизаторов. Однако в больших сетях, где состав сети часто изменяется, бывает неудобно назначать каждому компьютеру IP-адрес вручную. Во-первых, это отнимает время, а во-вторых, легко запутаться в большим количеством выданных IP-адресов. Также часто бывает, что в распоряжении предприятия (например, провайдера Интернет) имеется недостаточное количество IP-адресов, чтобы выделить каждому пользователю собственный IP-адрес, но (в связи с тем, что не все пользователи работают в сети одновременно) можно решить эту проблему, димнамически выделяя IP-адреса только тем пользователям, которые подключаются к сети в данный момент. Для динамического назначения IP-адресов используется служба DHCP. При подключении к сети компьютера пользователя, он посылает запрос на DHCP-север (для этого на компьютере пользователя необходимо указать Мое сетевое окружение/Свойства/

Протокол TCP/IP /Свойства /Получить IP-адрес автоматически). DHCP-сервер ищет в своей базе свободный в данный момент IP-адрес, и передает его клиенту вместе с другими настройками сети (IP-адреса серверов DNS и WINS, маршрутизаторов и др.). Настройка DHCP-сервера происходит при помощи меню Пуск/Настройка/Панель управления/Администрирование/DHCP. При настройке указывают следующие сведения:

- Допустимые диапазоны IP-адресов для динамического назначения пользователям (пул адресов). Адреса, зарезервированные для ручного назначения. При помощи DHCP можно также постоянно назначать конкретному компьютеру (с определенным именем и MAC-адресом сетевой карты) один и тот же IP-адрес.
- Допустимые настройки сети (IP-адреса DNS и WINS серверов, маршрутизаторов).
- Продолжительность аренды, предоставляемой сервером. Аренда определяет промежуток времени, в течение которого назначенный IP-адрес может использоваться.

5.7. Маршрутизация и удаленный доступ

Меню Пуск/Настройка/Панель управления/Администрирование/Маршрутизация и удаленный доступ позволяет создать маршрутизатор локальной сети и сервер удаленного доступа к сети (RAS, Remote Access Server). Функции и принципы работы маршрутизатора, а также основные протоколы маршрутизации были рассмотрены ранее в лекциях. Сервер удаленного доступа позволяет организовать подключение удаленных пользователей к серверу (и, при желании, ко всей остальной сети) при помощи модема. Создание маршрутизатора и сервера удаленного доступа можно осуществить при помощи удобных мастеров (Выделить сервер / Контекстное меню/Настроить и включить маршрутизацию и удаленный доступ) или выбрать ручную настройку. Ниже приведен список некоторых задач и методы их решения.

Таблица 5.3.

Настройка маршрутизации и удаленного доступа

Задача	Решение
Просмотр или создание новых сетевых интерфейсов.	Выделить сервер / Интерфейсы маршрутизации.
Задание статических маршрутов вручную.	Выделить сервер/IP-маршрутизация/Статические маршруты/Контекстное меню/Новый статический маршрут
Задание протоколов маршрутизации.	Выделить сервер/IP-маршрутизация/Общие/Контекстное меню/Новый протокол маршрутизации. Можно воспользоваться протоколами RIPv2 и OSPF.
Задание приоритета использования маршрутной информации, поступающей из разных источников.	Если до одного и того же компьютера/сети в таблице маршрутизации существует несколько маршрутов, то они используются в следующем порядке: 1) статические маршруты, 2) маршруты, полученные по протоколу OSPF, 3) маршруты, полученные по протоколу RIP. Можно изменить приоритеты обработки маршрутной информации: Выделить сервер /IP-маршрутизация/Общие/Контекстное меню/ Свойства/Уровни предпочтений.
Отображение существующих маршрутов.	Выделить сервер/IP-маршрутизация/Статические маршруты/ Контекстное меню/ Отобразить таблицу IP-маршрутизации
Запрещение продвижения пакетов между сетевыми интерфейсами	Выделить сервер/Контекстное меню/Свойства/Вкладка "IP"/ Разрешить IP-маршрутизацию - снять флажок. Если компьютер используется как маршрутизатор локальной сети, то иногда запрещают продвижение пакетов между сетевыми интерфейсами, а доступ из одного сегмента сети в другой осуществляют при помощи прокси-сервисов, прослушивающих соответствующие порты (подробнее о прокси-серверах и их преимуществах, с точки зрения обеспечения безопасности, см. ранее в лекциях). Если компьютер используют в качестве сервера удаленного доступа, то запрет IP-маршрутизации позволит удаленным пользователям подключаться только к серверу, но не даст доступа к остальной сети.
Задание трансляции сетевых адресов (NAT)	Трансляция сетевых адресов NAT уже подробно рассматривалась ранее в лекциях и позволяет большому количеству компьютеров работать с Internet или другой сетью при помощи одного или нескольких IP-адресов. Использование NAT также позволяет скрывать структуру своей сети от внешней сети, т.к. для внешней сети вся внутренняя сеть будет представлена всего одним IP-адресом. Настройка NAT осуществляется через меню IP-маршрутизация/Общие/Контекстное меню/Новый протокол маршрутизации/NAT-преобразование сетевых адресов. После создания протокола NAT: IP-маршрутизация/NAT/Контекстное меню/Новый интерфейс/ Выбрать интерфейс, соответствующей внутренней сети и в диалоговом окне указать "Частный интерфейс, подключен к частной сети". Аналогично выбрать второй интерфейс, соответствующий внешней сети (например, Internet), и в диалоговом окне указать/Общий интерфейс, подключен к интернет, а также установить флажок "Преобразовать TCP/UDP заголовки" и на вкладке "Пул адресов" указать IP-адрес, которым ваша внутренняя сеть будет представлена во внешней сети. Можно указать

Задача	Решение
	несколько адресов, или при помощи кнопки "Резервирование" указать, что конкретный IP-адрес внутренней сети , всегда будет заменяться в пакетах на конкретный IP-адрес внешней сети.
Создание IP-туннеля	При создании IP-туннеля между двумя компьютерами устанавливается логическое (а не физическое) соединение точка-точка. Если между компьютерами А и В существует IP-туннель, и пакет был направлен в этот туннель, то он помещается в дополнительный IP-пакет, в котором в качестве адреса назначения будет указан компьютер В. После поступления такого IP-пакета на компьютер В, из него извлекается первоначальный пакет и передается далее по сети, к которой подключен компьютер В. Обычно IP-туннель (интерфейс IP-в-IP) используется для перенаправления многоадресного IP-трафика из одной части сети в другую, через участок сети, в котором не поддерживается многоадресный IP-трафик. Создать IP-туннель можно следующим образом: "Интерфейсы маршрутизации/ Контекстное меню/Создать IP-туннель", а затем "IP-маршрутизация /Общие/ Контекстное меню/Новый интерфейс/ Выделить созданный ранее IP-туннель/OK/в диалоговом окне задать локальный (компьютер А) и удаленный (компьютер В) IP-адрес. В том же диалоговом окне, на вкладке "Общие", можно установить фильтры на входящие и исходящие пакеты туннеля (фильтрация по IP-адресам, типам протоколов, портам).
Создание интерфейса вызова по требованию	Поясним на примере. Если известно, что доступ к сети 15.0.0.0 можно получить при помощи модема (адаптера ISDN, другого устройства), позвонив по тел. 555-00-15, а доступ к сети 17.0.0.0 можно получить при помощи модема, позвонив по тел. 555-00-17, то для автоматизации этого процесса можно создать два интерфейса вызова по требованию и добавить в таблицы маршрутизации записи, отправляющие пакеты до соответствующих сетей на эти интерфейсы. Тогда если на маршрутизатор попадет пакет до сети 15.0.0.0, то модем автоматически наберет номер 555-00-15, установит соединение с удаленным компьютером (маршрутизатором) и передаст пакет. В случае если появится пакет до сети 17.0.0.0, то соединение будет установлено по номеру 555-00-17. Настройка интерфейса вызова по требованию происходит следующим образом: Интерфейсы маршрутизации/ Контекстное меню/ Создать новый интерфейс вызова по требованию/Указать тел. и др. параметры. Затем: IP-маршрутизация/Статические маршруты/Контекстное меню/Новый статический маршрут/Выбрать созданный интерфейс вызова по требованию и указать IP-адрес и маску сети (компьютера) назначения. Может также понадобиться указать: "Выделить сервер/Контекстное меню/Свойства/Вкладка "Общие"/ Использовать компьютер как маршрутизатор локальной сети и вызова по требованию".
Настройка компьютера в качестве сервера удаленного доступа (RAS)	Выделить сервер/Контекстное меню/Свойства/Вкладка "Общие"/Использовать компьютер как сервер удаленного доступа – установить флажок. В том же диалоговом окне осуществляются и другие настройки: Вкладка "Безопасность" – указывается выбор между службами проверки подлинности и учета пользователей (служба Windows или служба Radius). Кнопка "Методы проверки подлинности" позволяет включить запрос пароля по схеме CHAP или PAP (подробнее см. ранее в лекциях, протокол PPP) или разрешить удаленное подключение без проверки пароля и имени пользователя. Вкладка "IP" – назначение IP-адресов для подключающихся пользователей: используя протокол DHCP или из статического пула адресов, задаваемого вручную на этой же вкладке (адреса DHCP, DNS и WINS серверов выбираются автоматически, см. выпадающий список "Адаптер"). В дополнение к этому, в политике безопасности удаленного доступа (см. ниже) указывается: назначает ли сервер IP-адрес клиенту, или клиент может сам запросить IP-адрес. Вкладка "PPP" – позволяет задавать объединение нескольких физических подключений (например, несколько модемов) в один логический канал, а также управлять пропускной способностью канала, создавая дополнительные подключения при необходимости. Помимо описанных выше процедур, необходимо также создать соответствующих пользователей (Пуск/Настройка/Панель управления/Администрирование/Active Directory – пользователи и компьютеры/Выделить подразделение/Создать/Пользователь), задать им пароли и разрешить для них "Входящие звонки" (Active Directory – пользователи и компьютеры/Выделить пользователя/Двойной щелчок/Входящие звонки).

Продолжение табл. 5.3

Задача	Решение
Настройка политики удаленного доступа	Создание политики – Выбрать сервер/Политика удаленного доступа/Контекстное меню/ Создать политику удаленного доступа. В уже созданной политике используя кнопку "Добавить", можно задать следующие условия, по которым пользователям будет разрешено/отказано в удаленном доступе: - номер телефона исходящего звонка, который набрал пользователь. - номер телефона входящего звонка. - используемые протоколы и тип службы, которые запрашивает пользователь. - IP-адрес пользователя. - время звонка пользователя. - группа пользователей, к которой принадлежит звонивший и др. Используя кнопку "Изменить профиль" можно настроить профиль подключаемого пользователя: - назначается ли сервером IP-адрес клиенту, или клиент может сам запросить IP-адрес. - задать фильтр пакетов (межсетевой экран, firewall) для данного подключения, ограничивающий прохождение пакетов от клиентов и к клиенту, в зависимости от типа протокола и номера портов. - ограничить максимальную продолжительность и время звонков. - определить методы проверки подлинности и шифрования и др.
Настройка ведения журналов событий удаленного доступа и маршрутизации	Для настройки регистрации всех событий, связанных с удаленным доступом и маршрутизацией следует выбрать меню "Выделить сервер/Контекстное меню/Свойства/Журнал событий" – позволяет указать условия записи событий сервера, связанных с маршрутизацией и удаленным доступом (записывать только ошибки/ошибки и предупреждения/все события/отключить запись). Для настройки ведения журнала удаленного доступа необходимо воспользоваться меню "Выбрать сервер/Ведение журнала удаленного доступа" – позволяет настроить размеры, местоположение, формат файла журнала удаленного доступа и уровень детализации ведения журнала.

5.8. Диспетчер служб Интернета IIS (Internet Information Services).

Internet Information Services (Пуск/Настройка/Панель управления/Администрирование/Диспетчер служб Интернета) позволяет настраивать и администрировать web-, ftp-, smtp- и nntp- (группы новостей) сервисы на машине. Из-за постоянных проблем с безопасностью рекомендуется не использовать IIS и даже не устанавливать его на компьютер (например, в качестве web-сервера лучше использовать Apache).

Создать Web-сервер можно следующим образом: Выделить сервер/Контекстное меню/Создать/Узел Web/Отвечать на вопросы мастера: указать имя узла, IP-адрес (содержимое Web-узла или отдельные каталоги могут находиться как на данном сервере, так и на других компьютерах в сети), порт, каталог, разрешения (чтение, запуск сценариев, выполнение CGI-приложений, запись, обзор). Аналогично создается узел ftp, виртуальный почтовый сервер SMTP и виртуальный сервер новостей SMTP.

Настройка сервисов осуществляется следующим образом: "Выделить Web-узел (ftp, smtp, nntp)/ Контекстное меню/Свойства/". Можно устанавливать домашний каталог сервиса и определить разрешения для него (чтение, запись, обзор каталога, доступ к тексту сценария, запись в журнал, индексация каталога), запретить доступ к web(ftp)-узлу с определенных IP- или URL-адресов, устанавливать времени отключения не отвечающего пользователя, предельное число подключенных пользователей, вести журнал подключений, разрешать или запрещать анонимное подключение, просматривать текущие подключения к серверу, настроить вид html-страниц, возвращаемых пользователю при возникновении ошибок, название html-страницы, отображаемой по умолчанию и т.д. В меню "Выделить сервер/Свойства" можно ограничить полосу пропускания для всех web- и ftp-узлов данного компьютера, ограничив нагрузку на сеть, например величиной 1024 Кбит/с.

Особенностью IIS является поддержка активных серверных страниц (Active Server Pages, ASP). ASP позволяет динамически формировать HTML-страницы. ASP-файл представляет из себя документ HTML, в текст которого включены команды сценария ASP. Перед выдачей ASP-файла клиенту web-сервер обрабатывает команды ASP-сценария и динамически формирует HTML-страницу. Языком написания ASP-сценариев является VBScript (хотя могут использоваться языки JavaScript и Perl). Команды ASP-сценария встраиваются в HTML-страницу при помощи тэгов <Script> </Script> или <% %>. Отличием ASP-сценариев от обычных сценариев на языке VBScript/JavaScript является то, что если обычные сценарии выполняются на стороне клиента, то команды ASP-сценария выполняются на стороне сервера и пользователь получает "готовый" HTML-документ без всяких тэгов <Script> </Script> (только если сам ASP-сценарий не сформировал новые тэги <Script>). То, что ASP-сценарий выполняется на стороне сервера значительно расширяет его возможности. Так, например, в HTML-страницу могут быть динамически вставлены сведенья

из базы данных, хранящейся тут же на сервере. Раньше (и до сих пор, на всех Unix/Linux системах) для динамического формирования HTML-страниц использовались CGI-программы – программы на языках C, Perl и др., удовлетворяющих Общему Шлюзовому Интерфейсу (Common Gateway Interface, CGI). ASP-сценарии призваны заменить CGI-программы и упростить создание динамических HTML-страниц. Однако из-за низкой популярности IIS и Windows 2000, в качестве сервера Internet, ASP-скрипты пока не получили столь же широкое распространение, как CGI-программы.

Примечание: если на компьютере установлен IIS 5.0, то достаточно полную справку по IIS и ASP можно получить, набрав в браузере Internet Explorer адрес http://localhost/iisHelp/iis/misc/default.asp. (как следует из адреса, подключение к Internet для этого не требуется :).

5.9. Служба Telnet.

Служба Telnet позволяет организовать подключение пользователей к серверу по протоколу Telnet. Telnet – это фактически протокол эмуляции терминала: каждый символ, введенный пользователем на своем компьютере, будет считаться символом, введенным на сервере. При подключении к серверу Telnet запускается оболочка – специальная программа, которой и будут передаваться нажатия клавиш пользователя. По умолчанию это программа вида C:\WINNT\System32\cmd.exe – командная строка. Используя командную строку, можно отдавать серверу стандартные команды, типа dir, cd, copy и т.д. В качестве оболочки можно использовать не только командную строку, но и любую другую программу, в том числе, написанную самостоятельно. Общим здесь остается только одно: все нажатия клавиш пользователя будут переданы оболочке.

Настройка службы Telnet осуществляется Пуск/Настройка/Панель управления/Администрирование/Управление сервером Telnet. Управление осуществляется посредством текстового меню, которое позволяет вывести список текущих пользователей, прервать сеанс пользователя, запустить/остановить службу Telnet и изменить ее конфигурацию (см. табл. 5.4).

Таблица 5.4.

Параметры сервера Telnet.

№	Название	Описание и допустимые значения	Стандартно
1	AllowTrusted Domain	0: Разрешен доступ только для локальных пользователей. 1: Разрешен доступ пользователям из доменов с доверительными отношениями.	1
2	AltKey Mapping	0: Ctrl-A воспринимается как Ctrl-A (работает только для VT100). 1: Ctrl-A воспринимается как Alt (работает только для VT100).	1
3	Default Domain	Домен по умолчанию. Можно указать любой домен с доверительными отношениями. Локальный домен обозначается символом "." (точка).	
4	DefaultShell	Задает оболочку. По умолчанию: %systemroot%\System32\Cmd.exe /q /k	
5	LogonScript	Задает путь к расположению сценария входа на сервер Telnet. Администратор может настроить сценарий входа на выполнение определенных функций для каждого пользователя. По умолчанию: %systemroot%\System32\login.cmd	
6	MaxFailed Logins	Задает максимальное число неудачных попыток входа в систему перед завершением подключения.	3
7	NTLM	0: Стандартная проверка подлинности пользователя. Имя пользователя и пароль передаются в открытом виде. По соображениям безопасности не рекомендуется использовать параметры 0 и 1, однако в таком случае может возникнуть проблема совместимости с клиентами Telnet других операционных систем. В таком случае лучше вообще отказаться от использования Telnet. 1: Сначала предпринимается попытка выполнить проверку подлинности NTLM. При сбое используется стандартная проверка (имя пользователя и пароль). 2: Используется только проверка подлинности NTLM. Работает только между компьютерам, работающими под управлением Windows NT или Windows 2000. Подробнее о NTLM см. в разделе "Система безопасности Windows 2000".	2
8	TelnetPort	Задает порт, на котором работает служба Telnet.	23

5.10. Диспетчер службы терминалов

Несколько пользователей сети, с удаленных компьютеров, могут одновременно подключиться к программам Windows, работающим на сервере, используя службу терминалов. При помощи оснастки "Пуск/Настройка/Панель управления/Администрирование/Создатель клиента службы терминалов" администратор создает установочные дискетты (2 для Windows 9x, 4 для Windows 3.11) и обходит с ними соответсвующие компьютеры, устанавливая и настраивая клиента. После установки клиента, пользователи могут работать на своих компьютерах так, как будто бы они сидели за клавиатурой сервера: все нажатия клавиш и движения мыши передаются на сервер, программы выполняются на сервере, используя его ресурсы, а пользователям передается соответствующее изображение экрана монитора, которое и отображается на компьютере пользова-

теля. Такая схема позволяет съэкономить средства: компьютеры пользователей могут не модернизироваться – все равно они будут использоваться только как терминал (монитор и клавиатура), а все программы будут реально выполняться на сервере. Это позволяет добиться большой "скорости работы" даже на 486-х компьютерах с 32 Мб памяти, при условии, что вы не съэкономили на оперативной памяти сервера. Для стандартных "офисных" программ необходимо минимум 100Мб для самой Windows 2000 плюс по 20Мб на каждого клиента. При использовании графических и "ресурсоемких" приложений, объем памяти необходимо скорректировать в большую сторону. Также полезно отключить неиспользуемые сервисы ("Пуск/Настройка/Панель управления/Администрирование/Сервисы").

Диспетчер службы терминалов ("Пуск/Настройка/ Панель управления/Администрирование/ Диспетчер службы терминалов") обеспечивает настройку терминала Windows 2000 Server, позволяет вывести список пользователей подключившихся через терминалы, подключить или отключить пользователя, отправить ему сообщение.

Единственное неудобство при работе с терминальным сервером – это лицензирование. Перед использованием терминального сервера необходимо активизировать "Лицензирование службы терминалов" и проинсталировать купленные лицензии (в виде дискеты или серийного номера). При подключении клиента, сервер терминальных лицензий проверяет, есть ли у клиента лицензия на подключение (Windows 2000 Professional всегда имеет встроенную лицензию). Если лицензии нет то сервер проверяет, есть ли у него в банке свободные еще не розданные лицензии. Если свободная лицензия есть то она автоматически выдается этому клиенту, причем лицензия выдается на конкретную машину и уже не может быть самостоятельно возвращена в банк или передана другой машине, а при переустановке Windows лицензия теряется. Если лицензии кончились то клиенту выдается временная лицензия на 90 дней, после чего терминальный клиент перестает работать. Это неудобно, однако легко обходится либо установкой на сервере фиксированной даты (используется редко и как временная мера), либо удалением каждые 89 дней у клиентов всего содержимого реестра по адресу: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSLicensing , после чего клиент получает новую лицензию на 90 дней. Последний способ является нарушением авторских прав корпорации Microsoft и не может использоваться в коммерческих целях.

5.11. Управление компьютером.

Меню "Пуск/Настройка/Панель управления/Администрирование/Управление компьютером" позволяет управлять локальным компьютером или другим компьютером в сети ("Управление компьютером/Контекстное меню/Подключиться к другому компьютеру"), при условии, что на нем установлена Windows 2000/NT и удаленное управление разрешено. При помощи команды "Управление компьютером/Контекстное меню/ Свойства/Дополнительно/Загрузка и восстановление" можно установить операционную систему загружаемую по умолчанию (если на компьютере установлена не только Windows 2000) и время, которое отображается меню выбора операционной системы. Другие команды меню кратко обобщены ниже.

Таблица 5.5.

Меню "Управление компьютером"

Раздел / подраздел	Пояснения
Раздел "Служебные программы"	
Просмотр событий	Позволяет просмотреть журналы событий системы (эта команда также доступна через меню Пуск/Настройка/Панель управления/Администрирование/Просмотр событий). Журналы событий подробнее будут рассмотрены в разделе "Система безопасности Windows 2000".
Сведения о системе	Позволяет просмотреть детальные сведения о системе, включая характеристики оборудования и установленного программного обеспечения.
Оповещения и журналы производительности	Позволяет настраивать средства наблюдения за производительностью системы.
Общие папки	Позволяет просмотреть общие ресурсы (диски/каталоги), доступные для сетевых пользователей, просмотреть/отключить текущие сеансы (подключившихся пользователей), просмотреть/принудительно закрыть все открытые по сети файлы.
Диспетчер устройств	Позволяет просматривать/отключать/включать/настраивать устройства компьютера (сетевые карты, модемы, мониторы, контролеры жестких дисков и т.д.). Именно здесь можно сменить драйверы устройств или устранить конфликты в распределении ресурсов системы между устройствами
Локальные пользователи и группы	Позволяет создавать пользователей/группы пользователей на локальном компьютере. Если на компьютере установлена служба каталогов Active Directory, то эта команда недоступна, а управление осуществляется через меню Active Directory – пользователи и компьютеры.

Продолжение табл. 5.5.

Раздел / подраздел	Пояснения
Раздел "Запоминающие устройства"	Позволяет управлять жесткими дисками (подробнее см. NTFS) и съемными носителями.
Раздел "Службы и приложения / Службы"	Позволяет отобразить запущенные на компьютере службы, просмотреть зависимости между службами, а также настроить режим их запуска (автоматически, вручную, отключено), определить учетную запись пользователя, от имени которой работает служба, определить поведение службы при первом, втором и последующих сбоях (ничего не делать, перезапустить службу, запустить определенную программу, перезагрузить компьютер). Доступ к настройке служб осуществляется через меню "Управление компьютером/Службы и приложения/Выделить службу/Свойства". Меню "Службы" также доступно через "Пуск/Настройка/Панель управления/ Администрирование/Службы".

5.12. Система безопасности Windows 2000

5.12.1. Пользователи и группы пользователей, права доступа, аудит.

Используя понятие "пользователь" и "группа пользователей" Windows 2000 позволяет реализовать достаточно мощную систему разграничения прав доступа. В Windows 95/98 запрос имени и пароля пользователя использовался только при работе в сети. При локальном входе на компьютер (без подключения к сети) пароль не требовался и любой человек мог изменить/ удалить любой файл, запустить/остановить любую программу, заразить компьютер вирусом и т.д. В Windows 2000 такое невозможно: при входе в систему пользователь в обязательном порядке указывает свое имя и пароль. Если имя и пароль верны, то пользователь входит в систему и получает строго ограниченный набор прав. Он может выполнять только те действия, которые разрешил ему администратор. При этом можно настроить аудит – регистрацию обращений пользователей к тем или иным объектам (файлы, принтеры и т.д.).

Создание пользователя осуществляется при помощи меню "Пуск/Настройка/Панель управления/ Администрирование/Active Directory – пользователи и компьютеры" (если Active Directory не установлена, то управление осуществляется через "Пуск/Настройка/Панель управления/Администрирование/Управление компьютером/Служебные программы/Локальные пользователи и группы"). Для создания пользователя необходимо выделить то подразделение, где он будет создан (или папку Users) и дать команду "Контекстное меню/Создать/Пользователь". Группа пользователей создается при помощи команды " Контекстное меню/ Создать/Группа". Группы используются для более удобного управления правами доступа – нет необходимости вручную определять права доступа для каждого члена группы. Права определяются в целом на группу, а члены группы получают эти права автоматически. Например, если какие-либо файлы доступны группе "Служба сбыта", то достаточно поместить сотрудника в эту группу, и файлы станут автоматически доступны и ему. Один и тот же сотрудник может быть членом в нескольких группах. Для того, чтобы изменить членство в группах необходимо воспользоваться командой "Выделить пользователя/Двойной щелчок/Член групп/Добавить". В этом же диалоговом окне, на вкладке "Учетная запись", можно запретить смену пароля пользователем, указать срок действия пароля, ограничить время входа пользователя и др. Вкладка "Среда" позволяет запустить определенную программу при входе пользователя в систему. Вкладка "Входящие звонки" позволяет разрешить/запретить входящие звонки от имени этого пользователя, определить номер, по которому перезвонит сервер удаленного доступа, при попытке пользователя подключиться к нему. Другие вкладки позволяют разрешить удаленное управление сеансом пользователя, указать адрес пользователя, номер телефона и пейджера, определить перемещаемый профиль пользователя, сценарии входа пользователя в систему и др.

Добавление пользователя в группу можно осуществить и другим способом: "Выделить группу/ Двойной щелчок/Члены группы/Добавить" (на этой же вкладке удобно просматривать список пользователей, входящих в данную группу). Можно сделать саму группу членом другой группы: "Выделить группу/ Двойной щелчок/Член групп/Добавить". Для быстрого добавления большого количества пользователей в группу можно воспользоваться командой "Выделить пользователей или подразделения/Контекстное меню/Добавить участников в группу".

Смена пароля пользователя может быть осуществлена, как самим пользователем ("CTRL-ALT-DELETE / Смена пароля"), так и администратором ("Выделить пользователя/Контекстное меню/Смена пароля"). Для того, чтобы временно отключить учетную запись (имя – пароль) пользователя, необходимо воспользоваться командой "Выделить пользователя/Контекстное меню/Отключить учетную запись". Для того, чтобы переместить пользователя в новое подразделение, необходимо воспользоваться командой "Выделить пользователя/Контекстное меню/Переместить". Подразделение – это аналог отдела предприятия, способ объединения пользователей и групп пользователей, со сходными требованиями к политике безопасности (например, подразделение "Бухгалтерия"). Для того, чтобы создать подразделение необходимо воспользоваться командой "Выделить имя домена/Контекстное меню/Создать/Подразделение". В рамках подразделения можно создавать пользователей и группы пользователей. Можно создавать подразделения,

вложенные в другие подразделения. Для подразделения можно задать групповую политику безопасности "Выделить подразделение/Контекстное меню/Свойства/Групповая политика/Создать" (подробнее групповая и другие политики безопасности будут рассмотрены в разделе "Политики безопасности Windows 2000").

Каждое имя пользователя/группы связывается с уникальным идентификатором пользователя/группы (например, S-1-2-34-56789012345-67890123456-7890123456-7890). Разграничение доступа и система аудита в Windows 2000 использует уникальные идентификаторы для назначения прав доступа и контроля доступа к разным объектам (файлы, принтеры и т.д.). При этом администраторы работают по прежнему с понятными именами пользователей, а уникальные идентификаторы используются только внутри самой Windows 2000. Невозможно в рамках одной лекции полностью описать всю систему разграничения доступа Windows 2000, поэтому приведем лишь несколько примеров разграничения доступа на основании имен пользователей/групп пользователей:

1) Для ограничения доступа к файлу/каталогу достаточно воспользоваться командой "Выделить файл (каталог)/Контекстное меню/Свойства/Безопасность" – указать пользователей/группы пользователей, имеющих доступ к данному файлу(каталогу), указать права этих пользователей. Нажав кнопки "Дополнительно" и "Показать/Изменить" на этой же вкладке, можно перейти к расширенному управлению правами доступа. В этом же диалоговом окне, на вкладке "Аудит", можно задать аудит (контроль) действий пользователя по доступу к этому файлу/каталогу. Можно настроить запись в специальный журнал (Пуск/Панель управления/ Администрирование/Просмотр событий/Журнал безопасности) успех или неудачу при попытке получить доступ к файлу (каталогу). Регистрация успешных попыток позволяет следить за разрешенной деятельностью пользователей, регистрация неудачных попыток – выявить попытки нарушения прав доступа.

2) Для ограничения доступа к различным разделам реестра, необходимо воспользоваться программой regedt32 (Пуск/Выполнить/regedt32).

3) Политики безопасности (см. ниже) могут запретить пользователю локальный вход на конкретный компьютер или/и доступ к конкретному компьютеру по сети. Соответствующие параметры политик называются "Локальный вход в систему" и "Отказ в доступе к компьютеру из сети" и доступны через меню "Пуск/Настройка/Панель управления/Администрирование/ Локальная политика безопасности/Параметры безопасности/Локальные политики/Назначение прав пользователя" (см. также политику безопасности домена, контролера домена, групповые политики – подробнее далее в лекциях).

5.12.2. Домены в Windows 2000, доверительные отношения между доменами, аутентификация пользователя (протоколы Kerberos и NTLM).

В Windows 2000 вся сеть подразделяется на домены. Домен - это семейство компьютеров, объединенных по некоторому признаку (например, домен "Бухгалтерия", домен "Маркетинг). Домен создается при установке Active Directory на сервер - контролер домена ("Пуск/Настройка/Панель управления/Администрирование/Настройка сервера/Active Directory"). В каждом домене свой набор рабочих групп и список пользователей. Домены имеют древовидную структуру, могут иметь подчиненные домены и подразделения.

В каждом домене назначается свой администратор. Администратор домена имеет право устанавливать права доступа только в своем домене. Таким образом, административные права не сконцентрированы в одних руках одного пользователя (как, например, root в Unix/Linux системах), а распределены между администраторами доменов, каждый из которых отвечает только за свою область. В рамках своего домена, администратор, путем установки дополнительных разрешений, может делегировать отдельным пользователям часть прав по администрированию домена, снимая с себя лишнюю работу. Например отдельному пользователю/группе пользователей можно делегировать следующие права по управлению подразделением "бухгалтерия": чтение всей информации о пользователе, создание, удаление и изменение информации в учетных записях пользователей (групп пользователей), изменение членства пользователя в группах, смена паролей пользователей, применение к подразделению той или иной групповой политики. Для этого достаточно отдать команду "Пуск/Панель управления/Администрирование/Active Directory – пользователи и компьютеры/ Выделить подразделение/Контекстное меню/Делегирование управления". Аналогично выполняется делегирование управление и для других объектов Active Directory: домена, стандартных пользователей (папка "Users"), компьютеров (папка "Computers") и контролеров домена ("Domain controlers").

Между доменами могут устанавливаться доверительные отношения ("Пуск/Панель управления/Администрирование/Active Directory – домены и доверие/Выделить домен/ Контекстное меню/ Свойства/Доверия"). Доверительные отношения означают, что если пользователь прошел регистрацию в домене "Бухгалтерия" (правильно ввел имя и пароль), и между доменом "Бухгалтерия" и доменом "Маркетинг" есть доверительные отношения, то пользователь может получить доступ к компьютерам домена "Маркетинг", не проходя там повторную регистрацию. Необходимо отметить, что пользователь домена "Бухгалтерия" даже не имеет в домене "Маркетинг" учетной записи (имя-пароль) и тем не менее может получить доступ к компьютерам "Маркетинга", естественно, в пределах прав, которые администратор домена "Маркетинг" установит для "людей из бухгалтерии". Доверительные отношения между доменами могут быть двусторонними (если А доверяет В, то и В доверяет А) или односторонними (если А доверяет В, то это не означает, что В доверяет А). Двусторонние доверительные отношения являются транзитивными,

т.е. если A доверяет B и B доверяет C, то и A доверяет C. Однако в любом случае, взаимодействие доменов в доверительных отношениях происходят только по криптографически защищенным протоколам Kerberos или NTLM, во избежание подмены злоумышленником одной из сторон доверительного взаимодействия.

Протоколы Kerberos и NTLM используются для аутентификации (подтверждении личности) пользователя. Протокол NTLM использовался еще в Windows NT 4.0, а протокол Kerberos появился в Windows 2000. Ниже кратко приведены сведения по этим протоколам:

- **протокол Kerberos V5** – специальный протокол, который подтверждает подлинность как пользователя, так и сетевых служб. Механизм таков: пользователь входит в сеть, введя свой пароль (система challenge-response) или используя смарт-карту. При правильности введенных данных Kerberos выдает пользователю "билет пользователя" (TGT) на все время нахождения в сети. Имея "билет пользователя", пользователь в любое время может обратиться к Kerberos, для получения "билета службы" (TGS - например, билет для почтовой службы или билет для доверенного домена "Маркетинг"), который во-первых подтверждает для службы подлинность пользователя (содержит зашифрованные данные о пользователе), а во-вторых подтверждает для пользователя подлинность службы, т.к. только подлинная служба (криптографически стойко подключенная к центру распространения ключей шифрования Kerberos) сможет правильно обработать зашифрованный "билет службы" и правильно ответить пользователю. Для самого пользователя весь этот процесс протекает абсолютно прозрачно: он только вводит свое имя и пароль при подключении к сети.
- **аутентификация NTLM** – используется для совместимости с предыдущей версией Windows NT 4.0. Система challenge-response: пользователь передает серверу свое имя, сервер возвращает пользователю случайное число, при помощи которого пользователь шифрует свой пароль (однонаправленная хэш-функция MD5) и возвращает его серверу, который имея в своей базе подлинный пароль пользователя, выполняет над ним туже операцию хэширования и сравнивает результат с пришедшим от пользователя. При совпадении – пользователь регистрируется. Такая методика позволяет избежать передачи пароля по сети в открытом виде. Причем каждый раз передается разное значение, перехват которого ничего не даст. Получить пароль по перехваченному хэш-значению, даже зная хэш-функцию – трудноразрешаемая задача.

5.12.3. Политики безопасности Windows 2000

Windows 2000 позволяет использовать достаточно большое количество политик безопасности для централизованного управления доступом. Политика безопасности – это набор стандартных правил, применяемых к группе пользователей (подразделению, домену, компьютеру) и описывающая единые требования к безопасности. Ниже приведен краткий обзор политик безопасности Windows 2000.

Таблица 5.6.

Виды политик безопасности

Вид политики	Область действия	Пункт меню
Политика безопасности контролера домена.	Определяет политику безопасности компьютера, являющегося контролером домена.	Пуск/Настройка/Панель управления /Администрирование/ Политика безопасности контролера домена
Политика безопасности домена.	Определяет общие установки политики безопасности всех компьютеров, входящих в домен.	Пуск/Настройка/Панель управления /Администрирование/Политика безопасности домена
Локальная политика безопасности.	Позволяет переопределить общие установки политики безопасности домена для конкретного локального компьютера.	Пуск/Настройка/Панель управления /Администрирование/Локальная политика безопасности
Групповая политика безопасности подразделения.	Позволяет определить установки политики безопасности для подразделения. Подразделение – это способ объединения пользователей и групп пользователей, со сходными требованиями к политике безопасности. Например, подразделение "Бухгалтерия".	Шаг 1. Создать подразделение предприятия. Пуск/Настройка/Панель управления /Администрирование/ActiveDirectory пользователи и компьютеры / Контекстное меню / Создать / Подразделение Шаг 2. Задать групповую политику для подразделения. Выбрать нужное подразделение / Контекстное меню / Свойства/ Групповая политика / Создать *для ссылки на уже ранее описанную групповую политику выбрать команду "Добавить".

Продолжение табл. 5.6.

Вид политики	Область действия	Пункт меню
Групповая политика безопасности домена.	Позволяет определить большое количество установок политики безопасности для домена. Фактически "Политика безопасности домена" является подразделом "Конфигурация компьютера/Конфигурация Windows/Параметры безопасности" данной групповой политики.	Пуск/Настройка/Панель управления /Администрирование/ActiveDirectory пользователи и компьютеры / Выделить нужный домен/Контекстное меню/Свойства/ Групповая политика/Создать.
Политика удаленного доступа	Позволяет ограничить доступ удаленных пользователей к серверу, через модем (подробнее см. ранее, в разделе "Маршрутизация и удаленный доступ).	Пуск/Настройка/Панель управления /Администрирование/Маршрутизация и удаленный доступ к сети/ Выбрать сервер/Политика удаленного доступа.

Таблица 5.7.

Политика безопасности контролера домена, политика безопасности домена, локальная политика безопасности – раздел параметры безопасности

Название раздела	Примеры параметров, пояснения
Политики учетных записей	
1.Политика паролей	Максимальный (минимальный) срок действия пароля, минимальная длина пароля, требовать неповторяемость паролей (система помнит N последних паролей и запрещает использовать их).
2. Политика блокировки учетной записи	Блокировка учетной записи (на N минут в случае неверного ввода пароля), пороговое значение блокировки (максимально допустимое число раз неверного ввода пароля), сброс счетчика блокировки (счетчика неверно введенных паролей) через N минут.
3. Политика Kerberos	Максимальный срок жизни билета пользователя (билета службы). Билеты криптографически надежно удостоверяют подлинность пользователей и служб в домене. Подробнее о системе аутентификации Kerberos см. далее в лекциях.
Локальные политики	
1. Политика аудита	Параметры раздела включают (выключают) запись в журнал безопасности системы неуспешные (успешные) попытки входа в систему, доступа к объектам, доступа к службе каталогов, управления учетными записями пользователей, аудит изменения политики безопасности, аудит системных событий и отслеживания процессов.
2. Назначение прав пользователя	Определить пользователей (группы пользователей), которым разрешен (запрещен): доступ к компьютерам домена (данному компьютеру) из сети, локальный вход в систему непосредственно на этом компьютере, завершение работы компьютера, изменение системного времени, архивирование и восстановление файлов и каталогов, увеличение дисковых квот пользователям, управление аудитом и журналом безопасности, делегирование части административных функций другим пользователям, получение прав владельца объекта, добавление новых компьютеров к домену и др.
3. Параметры безопасности	Позволяет потребовать использование безопасного канала передачи данных в сети (шифрование и цифровая подпись), задать сообщение для пользователей при их входе в систему, отключить обязательное нажатие CTRL+ALT+DEL перед входом в систему (не рекомендуется по соображениям безопасности), запретить пользователям установку драйверов принтеров, определить поведение системы при установке неподписанных программ и драйверов, потребовать обязательную очистку файла виртуальной памяти при выключении системы, задать уровень проверки подлинности пользователей (для совместимости с NT 4.0), переименовать стандартную учетную запись гостя и администратора и др.
Журнал событий	
1. Настройка протоколирования*	Максимальный размер журнала безопасности или количество дней, в течении которых хранятся события в журнале. Ограничить доступ гостей к журналу.

Продолжение табл. 5.7.

Название раздела	Примеры параметров, пояснения
Прочие	
1. Группы с ограниченным доступом*	Если занести группу пользователей в этот раздел, а затем двойным щелчком по группе открыть диалоговое окно и добавить пользователей в раздел “Члены этой группы”, то при перезагрузке системы и применении политики безопасности, только пользователи явно указанные посредством раздела "Группы с ограниченным доступом" будут сохранены в данной группе, остальные пользователи будут автоматически удалены из группы. Этот способ фактически дублирует обычные методы занесения пользователей в те или иные группы, однако здесь ключевым моментом является то, что этот способ интегрирован в политику безопасности и позволяет централизованно и наглядно проконтролировать членство в наиболее важных группах, со значительным объемом прав на систему (например, Администраторы).
2. Системные службы*	Позволяет задавать ограничения на режим запуска (вручную/автоматически/запрещено) системных служб, которые отвечают за определенные сервисы, например сервис Telnet. Определяет разрешения для определенных групп пользователей (пуск, остановка, запуск, удаление сервиса, смена/чтение разрешений, смена владельца, определение зависящих сервисов, опрос сервиса и т.д.), настроить аудит (для всех пользователей или отдельных групп пользователей можно задать какие действия – см. разрешения – будут записываться в журнал безопасности в случае их успеха/неуспеха).
3. Реестр*	Позволяет ограничить доступ к отдельным разделам реестра различным пользователям и группам пользователей. Также позволяет организовать аудит успешных/неуспешных действий по доступу к реестру для всех пользователей или отдельных пользователей и их групп.
4. Файловая система*	Позволяет ограничить доступ различных пользователей и групп пользователей к отдельным папкам и файлам. Также позволяет настроить аудит по результатам успешного/неуспешного доступа различных групп пользователей к отдельным папкам и файлам. Эти же операции можно выполнить и из "Проводника" (выбрать папку/файл, контекстное меню/свойства/Безопасность), однако данная настройка политики безопасности позволяет вести легко контролируемый единый список всех важных папок/файлов, а также позволяет запретить изменение разрешений и тогда только пользователи, имеющие право изменять политику безопасности, смогут изменить разрешения для этой папки.
5. Политики открытого ключа	Политика открытого ключа строится на основании асимметричных алгоритмов шифрования. Если в симметричных алгоритмах данные шифруются и дешифруются при помощи одного и того же ключа, то в асимметричных алгоритмах ключи создаются парами: закрытый ключ – открытый ключ. Данные, зашифрованные при помощи одного из этих ключей, не могут быть дешифрованы этим же ключом – для дешифровки необходим второй ключ. Такая схема позволяет организовать безопасное распространение ключей при котором нет необходимости скрывать открытый ключ: закрытый ключ остается у владельца, открытый – передается всем желающим. Любой человек может направить зашифрованное сообщение адресату, используя его открытый ключ. При этом другие пользователи, также имеющие открытый ключ адресата, не смогут прочитать сообщение, т.к. у них нет закрытого ключа. Более того, такая схема позволяет организовать цифровую подпись документов и программ: по тексту документа вычисляется необратимая хэш-функция (функция вида Y=f(X), в которой по значению Y нельзя получить значение X) и генерируется дайджест сообщения – обычно 128-битное число, результат вычисления хэш-функции. Если в тексте документа изменить хотя бы один бит, то его дайджест не совпадет с дайджестом, приложенным к документу. Чтобы приложенный к документу дайджест нельзя было подменить, он шифруется при помощи закрытого ключа отправителя. Любой человек, имеющий открытый ключ отправителя, может проверить корректность дайджеста, но только отправитель, имеющий закрытый ключ, может сформировать корректный дайджест. В Windows 2000 асимметричные алгоритмы реализованы посредством механизма сертификатов. Сертификаты — это своего рода электронные удостоверения пользователя (компьютера, службы), подписанные цифровой подписью

Продолжение табл. 5.7.

Название раздела	Примеры параметров, пояснения
	центра сертификации, в которых указывается данные о пользователе (компьютере, службе), открытый ключ пользователя, дата начала и окончания действия сертификата. Сертификаты используются для проверки подлинности сервера/клиента, защиты от изменений содержимого электронной почты или кода программ, установки криптографически защищенных штампов времени в документах, шифровании и восстановлении файлов шифрованной файловой системы EFS, шифровании IP-трафика и т.д. Управление центром сертификации осуществляется при помощи меню "Пуск/Настройка/Панель управления/Администрирование/Центр сертификации". Настройка "Политики открытого ключа" позволяет указать доверенные центры сертификации и список доверенных сертификатов, указать параметры автоматического запроса сертификатов. Она также позволяет создать агента восстановления шифрованной файловой системы EFS - пользователя, который может расшифровывать зашифрованные файлы других пользователей.
6. Политики безопасности IP	Позволяет задать параметры безопасности для IP-соединений: разрешить соединение, запретить соединение или потребовать определенный метод проверки личности пользователя (например, Kerberos), или алгоритм шифрования и проверки целостности данных и адресов IP-пакетов. Требования устанавливаются в зависимости от типа подключения (по локальной сети или через удаленный доступ), IP- или DNS-адресов, типа протокола, и портов компьютеров, участвующих в соединении. Фактически, настройка "Политики безопасности IP" реализуют простой, но достаточно гибкий межсетевой экран (firewall).

* - отсутствует в локальной политике безопасности.

Таблица 5.8.

Групповая политика безопасности (подразделения/домена)

Название раздела	Примеры параметров, пояснения
1. Конфигурация компьютера	
1.1.Конфигурация программ/ Установка программ	Позволяет автоматически установить необходимые программы всем пользователям в подразделении/домене. При этом на каждом конкретном компьютере программа только появляется в меню "Пуск" и реестре, а фактически устанавливается только в тот момент, когда пользователь в первый раз запускает программу.
1.2. Конфигурация Windows	
Сценарии (запуск, завершение)	Позволяет задать сценарии (exe-, bat-, vba-, js-файлы), выполняющиеся на компьютерах пользователей подразделения/домена (должна быть установлена Windows 2000) при запуске/выключении компьютера.
Параметры безопасности	Фактически представляет собой еще один способ обратиться к меню "Политика безопасности домена" (см. выше).
1.3. Административные шаблоны/Компоненты Windows	
NetMeeting	Позволяет запретить/разрешить пользователям подразделения/домена удаленное управление рабочим столом при помощи NetMeeting. Подробнее см. раздел "Конфигурация пользователя/Конфигурация Windows".
Internet Explorer	Позволяет установить общие для всех пользователей подразделения/домена настройки зон безопасности Internet Explorer и настройку разрешений зон безопасности (настройка зон безопасности и разрешений для этих зон выполняется в разделе "Конфигурация пользователя/Конфигурация Windows/Поддержка Internet Explorer). Пользователям будет запрещено самостоятельно изменять эти настройки. Можно также определить обязательные настройки использования прокси-сервера. Подробнее см. раздел "Конфигурация пользователя/Конфигурация Windows".
Планировщик заданий	Планировщик заданий отвечает за запуск по расписанию определенных программ. Можно запретить пользователям подразделения/домена просматривать/создавать /удалять задачи, останавливать запущенные задачи до их полного завершения и т.д.
Установщик Windows	Позволяет определить обязательные параметры установщика Windows (установка программ) или вообще запретить его использование пользователями подразделения/ домена. В этом случае программное обеспечение смогут устанавливать только администраторы.
1.4. Административные шаблоны/Система для пользователей подразделения/домена позволяет определить список программ, запускающихся автоматически, при входе в систему, или вообще отключить автозапуск. Можно отключить автоматическое шифрование для файлов, перемещаемых в зашифрованные папки и др.	
Вход в систему	Для пользователей подразделения/домена можно задать синхронный запуск сценариев входа в систему (рабочий стол не будет создан, пока не будут выполнены все сценарии) и асинхронный/синхронный режим выполнения сценариев (все сценарии выполняются параллельно/последовательно). Можно также определить максимальное время выполнение сценариев и режим отображения команд сценария, принудительно завершать сеанс пользователя при ошибке в перемещаемом профиле.
Дисковые квоты	Позволяет настроить обязательные параметры использования дисковых квот для пользователей подразделения/домена. Пользователи, не имеющие право изменять политику безопасности, не смогут самостоятельно изменить эти параметры.
DNS-клиент	Позволяет принудительно определить основной DNS-суффикс для пользователей подразделения/домена.
Групповая политика	Задает параметры использования групповой политики: интервал и режим обновления политики (применения новых параметров политики), асинхронное применение политики (ускоряет загрузку, однако возможно формирование рабочего стола еще до полного применения всех параметров политики) и др.
Защита файлов в Windows	Защита файлов Windows проверяет системные файлы Windows на наличие в них изменений. По умолчанию файлы сканируются только при установке программ. Использование этой политики позволяет задать сканирование системных файлов во время каждой загрузки системы.
1.5. Административные шаблоны/Сеть	
Автономные файлы	Windows 2000 позволяет кэшировать сетевые файлы на локальном компьютере для того, чтобы они были доступны даже при отключении сети. Данная политика разрешает/запрещает использование автономных файлов, а также настраивает параметры их использования.

Продолжение табл. 5.8.

Название раздела	Примеры параметров, пояснения
Сеть и удаленный доступ к сети	Windows 2000 позволяет организовать для локальной сети общий доступ к удаленной сети (например, Internet) через одно модемное подключение. Данная политика позволяет запретить администраторам, не имеющим право изменять данную политику, настраивать общий доступ. Подробнее см. раздел "Конфигурация пользователя/ Административные шаблоны/Сеть/Сеть и удаленный доступ к сети".
1.6. Административные шаблоны/ Принтеры	Позволяет настроить режим публикации (объявления) и обзора (просмотра) сетевых принтеров и др. параметры.
2. Конфигурация пользователя	
2.1.Конфигурация программ/ Установка программ	Аналогично разделу 1.1. (см. выше). Параметры, указанные в разделе "Конфигурация пользователя", частично совпадают с параметрами, указанными в разделе "Конфигурация компьютера". В таком случае параметры раздела "Конфигурация компьютера" имеют более высокий приоритет. Однако стоит проверять разделы политик с совпадающими названиями, т.к. чаще всего содержимое этих разделов не дублирует, а дополняет друг-друга.
2.2. Конфигурация Windows	
Поддержка Internet Explorer	Позволяет задать для пользователей подразделения/домена внешний вид Internet Explorer (заголовки окна, фон и кнопки панели инструментов, содержимое меню "Избранное", адрес домашней страницы), параметры подключения к Internet (например, прокси-сервер), настройки зон безопасности и разрешений для этих зон и др. параметры.
Сценарии входа/ выхода из системы	Позволяет задать для пользователей подразделения/домена сценарии, выполняющиеся при входе/выходе пользователя из системы. Эти сценарии являются общими для всех пользователей подразделения/домена.
Параметры безопасности	Подраздел Политики открытого ключа/Доверительные отношения позволяет создать список доверия сертификатов для пользователей подразделения/ домена.
Службы удаленной установки	Позволяет настроить параметры установки для пользователей Подразделения/Домена: возможность выполнять выборочную установку, возможность автоматической установки, перезапуска установки и др. параметры.
Перенаправление папки	Позволяет задать для пользователей подразделения/домена расположение папок "Мои документы", "Главное меню", "Рабочий стол". Для различных пользователей/групп пользователей можно указать различное расположение, или указать общее расположение для всех пользователей.
2.3. Административные шаблоны / Компоненты Windows	
Net Meeting	Позволяет настроить обязательные для всех пользователей подразделения/домена параметры Net Meeting: запретить общий доступ к приложениям, командной строке, окнам проводника, рабочему столу, удаленное управление при помощи Net Meeting, скрыть страницы настройки, ограничить пропускную способность сети или вообще запретить передачу аудио- и видеоданных, запретить прием/передачу файлов, ограничить размер передаваемых файлов.
Internet Explorer	Позволяет полностью контролировать внешний вид Web-браузера Internet Explorer (содержимое меню, кнопки, вкладки, запретить изменение настроек языков, цветов, прокси-сервера и др.), ограничить максимальный размер получаемых файлов, ограничить использование Windows Media, ShockWave Flash и др. программ, связанных с Internet.
Проводник	Позволяет для пользователей подразделения/домена полностью контролировать внешний вид Проводника: содержимое меню, кнопки, запретить контекстное меню, скрыть значки "Мои документы", значки "Вся сеть", скрыть вкладку "Оборудование", скрыть некоторые диски локального компьютера, скрыть кнопку "Поиск", скрыть команды подключения/ отключения сетевых дисков, команды и др.
Консоль управления Microsoft	Позволяет для пользователей подразделения/домена ограничить возможность запуска большинства меню конфигурирования Windows, в том числе из папки "Администрирование".
Планировщик заданий	Аналогично такому же разделу в "Конфигурации компьютера".
Установщик Windows	Аналогично такому же разделу в "Конфигурации компьютера".

Продолжение табл. 5.8.

Название раздела	Примеры параметров, пояснения
2.4. Администра-тивные шаблоны / Панель задач и меню пуск	Позволяет для пользователей подразделения/домена настроить вид меню "Пуск" и панели задач: убрать из меню подменю "Документы", команды "Найти", "Выполнить", "Завершение работы", "Завершение сеанса", "Сеть и удаленный доступ к сети", запретить перетаскивание и контекстное меню в меню "Пуск" и на "Панели задач", запретить изменение параметров панели задач и меню "Пуск".
2.5. Администра-тивные шаблоны / Рабочий стол	Позволяет для пользователей подразделения/домена настроить вид рабочего стола: скрыть значки "Сетевое окружение", "Мои документы", запретить пользователям изменять положение папки "Мои документы", запретить изменение места положения панелей, скрыть все значки рабочего стола.
2.6. Административные шаблоны / Панель управления	
Установка и удале-ние программ	Позволяет для пользователей подразделения/домена запретить установку и удаление программ, скрыть некоторые пункты меню "Установка и удаление программ", запретить установку программ с CD-диска, дискет или по сети.
Экран	Позволяет для пользователей подразделения/домена запретить настройку экрана или отключить некоторые вкладки в меню настройки. Позволяет явно указать имя файла хранителя экрана для предотвращения проникновения в систему вредоносных программ через хранитель экрана (для этого необходимо не только указать имя файла-заставки, но и правильно выставить права доступа к этому файлу, чтобы пользователь не смог заместить его своим файлом, с таким же именем).
Принтеры	Позволяет запретить/разрешить установку или удаление принтеров, обзор принтеров по сети и др.
Язык и стандарты	Позволяет указать обязательный язык для меню и диалогов в Windows.
2.7. Административные шаблоны/Сеть	
Автономные файлы	Аналогично такому же разделу в "Конфигурации компьютера".
Сеть и удаленный доступ к сети	Для пользователей подразделения/домена позволяет ограничить доступ к настройке подключений по сети и удаленного доступа. Например, запретить дополнительную настройку TCP/IP (указание IP-адресов шлюзов, DNS и WINS серверов).
2.8. Административные шаблоны/Система для пользователей подразделения/домена позволяет скрыть средства редактирования реестра, запретить использование командной строки, выполнять только зарегистрированные приложения Windows, не запускать определенные программы (указывается список), отключить автозапуск или вообще задать особый интерфейс пользователя. По умолчанию запускается программа Explorer.exe, соответствующая стандартному интерфейсу Windows, но можно указать и собственную программу.	
Вход/выход из системы	Для пользователей подразделения/домена позволяет запускать указанные программы при входе в систему, запретить запуска диспетчера задач, блокировку компьютера, запретить изменение пароля, завершение сеанса и др.
Групповая политика	Задает параметры обновления групповой политики.

Лекция 6. Linux

К сожалению у меня не хватило времени переписать эту лекцию. Для самостоятельного изучения можно рассмотреть следующие вопросы:

Инсталляция Red Hat Linux 7.0. Использование пакетов RPM для инсталляции программного обеспечения в Linux Red Hat.

Принципы построения файловой системы Linux, файл /etc/fstab, структура каталогов, права доступа к файлам (команды chmod, chgrp, chown, umask), символические и жесткие ссылки (команда ln), монтирование и размонтирование файловых систем (команды mount и umount), создание и проверка файловых систем (команды mkfs и fsck), диспетчер файлов Midnight Comander.

Система безопасности Linux, пользователи (команда who, whoami), суперпользователь root, пароли пользователей и их затемнение (файлы /etc/passwd и /etc/shadow), проблема слабых паролей, добавление и удаление пользователей, временная блокировка пользователя, SUID-программы и проблемы безопасности.

Краткий перечень команд Linux для работы с файлами и файловыми системами: ls, cp, mv, rm, mkdir, rmdir, touch, cfdisk, df, dd, fdformat. Команды ввода-вывода: read, echo, cat, head, tail, more, page. Команды поиска и сравнения файлов: find, grep, cmp, diff. Архивация файлов: tar, gzip, gunzip.

Управление процессами и остановка системы: команды ps, kill, &, bg, fg, nohup, nice, shutdown, halt, reboot, ctrlaltdel. Команды справочной системы: man, apropos, info. Перенаправление ввода вывода, конвейеры команд, команда tee, named pipes. Псевдонимы команд (команда alias). Утилита awk и потоковй редактор sed.

Оболочки (shell) в Linux - bash и tcsh, написание скриптов на shell: переменные среды, использование кавычек в сценарии (двойные, одинарные, обратные), символы подстановки, escape-последовательности, управляющие конструкции (test, if then else, while, for, until, case), передача параметров сценарию, использование функций в сценарии.

Загрузка системы: менеджер загрузки LILO, процесс init (файл /etc/inittab, уровни выполнения runlevel, команда telinit), сценарии начальной загрузки (rc-скрипты), процесс getty, профили пользователя (файлы /etc/profile, etc/bashrc, $HOME/.bash_profile, $HOME/ .bash_logout, $HOME/ .bashrc). Выполнение команд по расписанию: планировщик cron (файлы crontab), команда at.

Настройка Linux при помощи пакета linuxconf. Внесение информации о сети в файлы /etc/hosts и /etc/networks, внесение информации о компьютере в файл /etc/issue.

Настройка IP-адресов сетевых интерфейсов и маршрутизации, фиктивный интерфейс, псевдонимы IP (команды ifconfig и route). Динамическая маршрутизация при помощи RIP (демон gated). Проверка работы сети при помощи команд ping, netstat, использование arp. Настройка подключения к Internet по протоколу PPP, автоматизация подключения при помощи chat, настройка подключения по требованию. Настройка ресолвера: указание порядка обращения к службам разрешения имен в файле /etc/hosts.conf . Службы разрешения имен: BIND (файл resolv.conf, демон named, программа nslookup) и NIS.

Удаленный вызов процедуры RPC: понятие, демон portmapper (файл /usr/sbin/rpc.portmap). Сетевая файловая система NFS, доступ к файловым системам Linux-сервера с рабочих станций Windows по протоколу SMB (настройка samba-сервера утилитой SWAT и вручную, запуск samba-сервера).

Настройка суперсервера служб Интернета – демоны inetd и tcpd (файлы inetd.conf , hosts.allow, hosts.deny), и их замена – суперсервер xinetd (файл xinetd.conf). Файлы служб и протоколов (/etc/services и /etc/protocols). Настройка web-сервера Apache httpd (файлы httpd.conf, access.conf, srm.conf, .htaccess) и ftp-сервера wu-ftpd, установка и настройка почтового сервера sendmail, настройка прокси сервера squid.

R-службы и их настройка (файлы hosts.equiv и .rhosts). Использование ssh для безопасного подключения к системе: настройка демона sshd и клиентов ssh (файл /etc/ssh/sshd_config, команда ssh-keygen и др.), использование ssh (команды slogin, scp, ssh), туннелирование прикладных протоколов через соединение ssh.

Межсетевой экран IPChains (ipfwadm, iptables), правила фильтрации, цепочки правил, пользовательские цепочки правил, учет IP трафика, маскировка IP и NAT.

Аудит в Linux: bash.history, демон syslogd (файл /etc/syslog.conf) и log-файлы др. служб. Управление log-файлами - команда logrotate (файл logrotate.conf). Проверка изменений в файлах системы при помощи программы tripwire.

Сетевая графическая система X-Window: сценарий startx, регистрация в системе через xdm, соединение с X-сервером (переменная DISPLAY, команда xhost, файл .Xauthority), графические рабочие среды Gnome и KDE. Инсталляция и использование пакета Star Office и др. ПО для Linux.

В будущем, лекция будет построена именно в соответствии с этим планом. Сейчас же, предлагаю для ознакомления старый, и далеко не самый полный вариант.

Linux - это один из клонов Unix, развившийся в самостоятельную операционную систему. Разработка ОС Linux выполнена Линусом Торвалдсом из университета Хельсинки. Отличительной чертой ОС Linux является то, что ее исходные тексты открыты и распространяются бесплатно (среди Unix-систем бесплатно распространяются тексты FreeBSD). Существует также множество коммерческих дистрибутивов (пакетов установки) Linux: Red Hat Linux, Mandrake Linux, Slackware Linux, Debian Linux, Corel Linux, Caldera OpenLinux, S.U.S.E. Linux, Black Cat Linux, Connectiva Linux и др.

Linux является надежной, эффективной и не требовательной к оборудованию многопользовательской, многозадачной операционной системой общего назначения, наилучшим образом подходящей для создания сервера Internet и использования в глобальных сетях TCP/IP. Linux является достаточно сложной и универсальной операционной системой, требующих профессиональных навыков для работы с ней. Только перечень команд Linux, с кратким описанием их параметров занимает больше 150 страниц. А для того, чтобы корректно настроить систему, необходимо тщательно изучить справку Linux (man-страницы) и специализированные руководства. И хотя в последнее время прилагаются значительные усилия, чтобы упростить использование Linux для обычных пользователей, но до сих пор Linux сохраняет статус удобной и эффективной операционной системы для профессионалов. Рядовому пользователю, не желающему тратить время на изучение и настройку операционной системы, для которого не столь важны производительность, надежность и сетевая безопасность, можно порекомендовать воспользоваться ОС Windows.

Возможности ОС Linux.

- обладает высоким быстродействием, работает надежно, устойчиво.
- эффективно управляет многозадачностью и приоритетами, фоновые задачи (длительный расчет, форматирование дискеты и т.д.) не мешают интерактивной работе;
- множественные виртуальные консоли: на одном дисплее несколько одновременных независимых сеансов работы, переключаемых с клавиатуры;
- графическая сетевая оконная система X Window (для Linux есть версия X Window, известная как XFree86; или версия X11R5).
- передовая файловая система объемом до 4 Терабайт и с именами файлов до 255 знаков, которая, в силу своей организации, мало подвержена вирусам;
- поддержка протоколов Internet (TCP/IP, поддержка ftp, telnet, NFS); работа с сетями на базе Novell и MS Windows;
- позволяет выполнять представленные в формате загрузки прикладные программы других ОС - различных версий Unix, DOS и MS Windows;
- доступ к дискам с файловыми системами в формате DOS, Windows, CD ROM (iso9660), hpfs.
- хорошо документирована, наличие исходного текста всех программ, включая тексты ядра, драйверов, средств разработки и приложений.

Графические рабочие среды, такие как KDE или Gnome делают использование Linux не сложнее, чем Windows, а Gnome еще и построена таким образом, чтобы максимально соответствовать Windows.

Оболочки Linux

При работе с командами Linux, пользователь чаще всего видит приглашение для ввода командной строки в виде " # " или " $ ". На самом деле, эти приглашения выдает не сам Linux, а оболочка - интерпретатор интерактивных команд Linux. Так например оболочки позволяют использовать в командах Linux символы подстановки, такие как * и ?. Кроме того, оболочка – это мощный командный язык, который позволяет писать программы (shell-scripts), объединяющие несколько команд в командный файл (аналог BAT-файлов в DOS). Две самые распространенные оболочки - это sh (shell Баурна) и csh (C shell). В Linux также используются bash (развитие sh) и tcsh (развитие csh). При работе с Linux пользователи фактически работают с одной из этих оболочек, однако они – не сам Linux, а лишь надстройки над ним.

Система X Window

Система X Window – это *сетевой оконный графический* интерфейс для Linux/Unix-машин, построенный на идеологии клиент-сервер. X-Window была разработана в Массачусетском технологическом институте (MIT). Используя X Window, пользователь может одновременно иметь на экране несколько окон, при этом каждое может выполняться от имени другого пользователя. В X-Window используется мышь, хотя она необязательна. Используя протоколы TCP/IP, вы можете по сети смотреть у себя содержимое X-окон, выполняющихся на других машинах. Интерфейс X Window в большой степени контролируется менеджером окон (например Open Look). Эта программа отвечает за размещение окон, изменение их размеров, перемещение окон, вид оконных рамок и т.д.

Файловая система Linux

В Linux все есть файл: принтер – файл, клавиатура, монитор или мышь – файл (/dev/console/, /dev/mouse), выполняющаяся в данный момент программа – файл. Например, вывод данных на принтер получается перенаправлением вывода информации в файл принтера, причем Linux не делает никакого отличия между файлом на диске и самим принтером. При работе в Linux необходимо учитывать, что она различает регистр символов и файлы myfail.txt и MyFail.TXT – это не одно и то же. Слэш - разделитель пути в каталогах Linux направлен в другую строну чем в Windows, т.е. не "path \ fail", а " path / fail". Более того, в Linux отсутствуют, привычные для пользователей DOS и Windows, диски A, B, C, D и т.д. Вместо этого, CD-ROM, гибкие и жесткие диски, подключаются как часть корневого каталога. При запуске компьютера сначала монтируется корневая файловая система, т.е. корневой каталог “/” (указанный при инсталляции Linux), а затем к нему монтируются все остальные жесткие диски и их разделы, указанные в файле /etc/fstab.

Таблица 6.1.

Структура каталогов Linux

Каталог	Пояснения
/	Корневой каталог. В Linux/Unix – системах слэш – в другую сторону, чем в MS DOS или Windows. Кстати, поскольку Internet – это исторически сеть Unix машин, то и адреса в Internet тоже с обратным слэшем.
/bin	Важные системные программы Linux, используемые при загрузке системы и обычными пользователями.
/sbin	То же, что и /bin, только находящиеся здесь команды не предназначены для пользователей с общими правами.
/etc	Конфигурационные файлы. Например, /etc/fstab – список подключаемых жестких дисков, /etc/rc – команды, выполняемые при запуске системы, /etc/passwd - файл паролей, /etc/shadow – теневая база паролей, /etc/group – информация о группах пользователей, /etc/securetty - терминалы, с которых может подключаться к системе пользователь root..
/usr	Каталог куда устанавливаются все программы пользователей.
/usr/etc	Файлы конфигурации несущественные для системы, но необходимые для пользовательских программ.
/usr/X11R6 /usr/X386	Файлы, используемые системой X Windows.
/usr/bin /usr/sbin	Практически все команды Linux не предназначенные для размещения в корневом каталоге (например, здесь находится большинство программ-серверов).
/usr/local	Отдельно устанавливаемые пакеты программ и другие файлы.
/root	Личный каталог пользователя root.
/home	Домашние каталоги пользователей. Например, /home/Ivan - домашний каталог пользователя Ivan
/mnt	Каталог куда обычно подключаются файловые системы: cdrom, дискетты, жесткие диски.
/dev	Файлы драйверов устройств. Они используются для доступа к устройствам и ресурсам системы, таким как диски, модемы, память и т.д. Например, имея доступ к файлу /dev/mouse вы можете читать входные сигналы от мыши, считывая данные из этого файла.
/proc	В действительности не существует на диске, а создается ядром ОС в памяти компьютера. Предоставляет информацию о системе (например /proc/meminfo - информация об использовании памяти) и выполняющихся программах. Так каталог /proc/1 содержит информацию о процессе номер 1 и т.д.
/boot	Файлы, используемые начальным загрузчиком ОС
/lib	Разделяемые библиотеки программ (аналог dll).
/var	Файлы, размер которых постоянно изменяется во время работы системы, такие как буферные каталоги (для почты, новостей и т.д.), журнальные файлы, страницы справки, а также временные файлы.
/tmp	Временные файлы.

Система безопасности Linux

ОС Linux является высоконадежной и устойчивой к повреждению вирусами. К Linux-машинам возможно удаленное подключение по протоколам HTTP, FTP, SMTP, TELNET, через механизм *демонов* (=серверы в Windows NT) – специальных программ, постоянно активных на Linux машине и позволяющих пользователю удаленно подключаться к ним. Все пользователи в Linux подразделяются на:

1) Суперпользователя – имеет неограниченные права и стандартное имя "root".
2) Обычный пользователь – имеет права с ограничениями, установленными суперпользователем ему и группе в которую входит пользователь.

3) Специальный пользователь – имеет дополнительные права для работы с конкретным приложением.
4) Псевдопользователь – пользователь, подключившийся к Linux-машине удаленно, через программу-демон. Не имеет никаких прав, не идентифицируется системой, все действия такого пользователя определяются возможностями программы-демона.
5) Владелец – пользователь создавший файл или каталог. Владелец имеет полный доступ к созданным им объектам, если он сам или root не установит дополнительные ограничения.

При подключении к Linux-системе пользователь вводит свой пароль. Пароли в зашифрованном виде хранятся на Linux-машине в специальном файле (/etc/passwd). Каждый подключившийся пользователь характеризуется своим уникальным идентификатором UID (User Identifier) и идентификатором группы GID (Group Identifier). Любой файл, созданный пользователем или запускаемая от его имени программа получают UID и GID пользователя. --> В Linux нет "ничьих" программ или файлов. Каждая программа может выполнять действия в пределах прав пользователя и его группы, а каждый файл может создаваться, читаться или изменяться только если у пользователя достаточно для этого прав. Таким образом всегда известен "автор" последних изменений в файле, а вирус, случайно принесенный пользователем , сможет разрушить только файлы пользователя и не сможет повредить системные файлы (нет прав) или файлы других пользователей. Такой подход обеспечивает достаточно стройную систему защиты, однако и в ней есть слабые места – во первых вирус принесенный пользователем root сможет повредить любые файлы, во-вторых катастрофической будет ситуация когда пароль root-а подберет какой-нибудь злобный вандал или сам root решит "насолить" начальству. Во-вторых, в Linux есть ряд программ, называемых SetUID (SUID/SGID)-программ, которые выполняются не от имени человека подключившегося к Linux, а от имени человека создавшего эти программы. Поясним на примере. Допустим, пользователю необходимо сменить собственный пароль подключения к Linux-машине. Естественно, пользователь должен иметь возможность сделать это, т.к. одно из основных правил безопасности – это периодическая смена паролей. Однако для смены пароля, пришлось бы разрешить пользователю чтение и запись в файл паролей (/etc/passwd), что недопустимо, т.к. в таком случае пользователь сможет его случайно испортить (и больше никто не получит доступ на Linux-машину) или сменить чужие пароли. Поэтому пользователю доступ к файлу паролей не разрешается. Вместо этого, системная команда смены пароля запускается с атрибутом SUID/SGID, т.е. не от имени пользователя, а от имени администратора root - владельца этой команды, создавшего ее при инсталляции Linux. Root имеет право чтения-записи в файл паролей, что позволяет пользователю изменить свой пароль, но только при помощи системной команды, созданной Root-ом. Любые другие программы пользователя доступа к файлу паролей не получат.

Краткий перечень наиболее употребимых команд Linux

Ниже будет приведен просто перечень базовых команд Linux. Более подробную информацию можно получить из справки Linux (команды справки также см. в списке команд):

Команды общего назначения

apropos	Поиск справки по ключевому слову
man, whatis	Поиск справки по точному названию команды Linux.
xman	Оконная графическая справочная система.
startx	Запуск графической оконной системы X-Window.
ls	Вывод оглавления каталога
pwd	Вывод имени текущего каталога.
cd	Смена текущего каталога.
mkdir	Создание каталога.
rmdir	Удаление каталога.
cp	Копирование файла.
mv	Перемещение файла.
rm	Удаление файла.
cat	Создание файла.
ln	Создание ярлыка.

find	Поиск файла.
grep	Поиск текста в файлах.
lp	Отправка файла на печать.
lpstat	Вывод информации о состоянии очереди печати.
cancel	Отмена печати.
ps	Вывод списка всех запущенных программ.
kill	Прерывание выполнения программы.
clear	Очистка экрана.
date	Текущая дата и время.
echo	Вывод текста на экран.
more	Вывод файла на экран по частям.
page	Постраничный вывод на экран.
sort	Сортировка.
crypt	Шифрование / дешифрование файла.
reboot	Перезагрузка компьютера.
halt	Выключение компьютера.

Администрирование

login	Вход систему, как определенный пользователь.
exit, logout	Завершение сеанса пользователя, выход из системы.
logname	Вывод имени текущего пользователя.
groups	Вывод списка групп, к которым принадлежит пользователь.
id	Вывод имени пользователя, его GID и UID.
passwd	Смена пароля пользователя.
chmod	Смена прав доступа к файлу, каталогу (чтение, запись, исполнение файла для владельца, членов группы владельца, прочих пользователей, всех пользователей).

chgrp	Смена группы, к которой принадлежит файл.
chown	Смена владельца файла.
who	Список пользователей, подключенных в данный момент к Linux-машине.
finger	Вывод подробной информации о конкретном пользователе, подключенном к Linux-машине: имя, UID/ GID, время работы и др.)
cron, crontab	Ежедневное (еженедельное, ежемесячное) выполнение программы.
at	Одноразовое выполнение программы в заданное время.

.

Лекция 7. Технологии глобальных сетей.

Как уже ранее упоминалось в лекциях, протяженность локальной сети ограничена по чисто техническим причинам. На сегодняшний день трудно подобрать пример, в котором диаметр локальной сети превысил несколько десятков километров. Хотя создавать локальные сети столь большой протяженности и возможно технически, но это не рационально экономически. Чаще всего, небольшие локальные сети удаленных офисов объединяются с центральной сетью предприятия, при помощи глобальных сетей (Wide Area Network, WAN). Самой широко известной глобальной сетью является Internet (сеть TCP/IP), однако существуют и другие глобальные сети, например сеть X.25. Ниже будут рассмотрены основные технологии глобальных сетей.

7.1. Выделенные и коммутируемые каналы – физическая основа построения глобальных сетей.

Особенностью глобальных сетей является большая протяженность линий связи, объединяющих локальные сети. Причем такие соединения являются соединениями типа "точка-точка", когда сетевой кабель используется для передачи информации только между двумя компьютерами (или другим сетевым оборудованием), соединенным этим кабелем. Существуют следующие типы каналов, используемых для соединения локальных сетей (или отдельного пользователя с локальной сетью):

- выделенная линия
- коммутируемая линия

При соединении по выделенной линии, связь между двумя сетевыми устройствами существует постоянно. В любой момент времени удаленный маршрутизатор (мост) может направлять пакеты в выделенный канал, не заботясь об установлении соединения. Использование выделенной линии для соединения локальных сетей – дорогостоящее решение, т.к. приходится платить за аренду линии, вне зависимости от ее фактического использования. Поэтому данный вариант оправдан, только если между сетями циркулируют большие объемы данных. Если же трафик невелик, то выгоднее использовать коммутируемую линию.

В коммутируемой линии связь с другим сетевым устройством (возможно с несколькими) устанавливается только при необходимости. При этом пользователь платит только за фактическое время соединения, однако на установление самого соединения тратится время, а также возможны отказы в установлении соединения по причине занятости линии. Частным случаем коммутируемой линии является соединение домашнего пользователя с провайдером Internet по телефонной линии, с использованием модема.

Существуют различные типы выделенных и коммутируемых линий: аналоговые телефонные линии, цифровые линии PDH (с интерфейсами T1/E1, T2/E2, T3/E3), цифровые линии SONET/SDH, цифровые линии ISDN (более точно: сети ISDN), асимметричные цифровые абонентские линии ADSL. Ниже будут кратко рассмотрены все эти технологии.

7.1.1. Аналоговые телефонные линии

Коммутируемые аналоговые телефонные линии наиболее часто встречаются при соединении домашних пользователей с Internet через модем (dial-up access). Для того чтобы линию, соединяющую Вас с другой сетью, можно было считать цифровой, все оборудование на пути от Вашей точки подключения, до точки подключения сетевого оборудования другой сети, должно быть цифровым. Для этого Ваш компьютер должен быть подключен к сети по цифровому абонентскому окончанию (DSL, digital serial line – общий термин для цифровых абонентских окончаний разных технологий). Даже если ваша АТС (телефонная станция) является цифровой, но вы подключены к ней при помощи обычного аналогового модема, то на ваше соединение с провайдером Internet будут накладываться ограничения аналоговой линии (скорость не более 33,6 Кбит/с в направлении "пользователь - АТС" и не более 56 Кбит/с в направлении "цифровая АТС - пользователь"). Более того, даже если вы подключились к цифровой АТС при помощи цифрового абонентского окончания, но ваш провайдер подключен к своей АТС по аналоговой линии, или любая АТС на пути от вас до провайдера не является цифровой, то ваше соединение с Internet также будет аналоговым.

Для работы на выделенных и коммутируемых аналоговых линиях используют модемы. Слово "модем" является сокращением от двух слов "модулятор" - "демодулятор". На передающем конце модулятор преобразовывает сигналы в форму, удобную для передачи по телефонной линии (с учетом полосы пропускания телефонной линии), а на приемном конце демодулятор осуществляет обратное преобразование сигналов (см. рис. 7.1). Передача данных одновременно ведется обеими сторонами во встречных направлениях (дуплексный режим работы). Модемы могут выполняться с 2-проводным (коммутируемые или выделенные аналоговые телефонные линии) и 4 -проводным окончанием (выделенные аналоговые линии). Модемы бывают внешними (отдельное устройство, соединяемое кабелем с COM-портом компьютера) или внутренними (плата, устанавливаемая в компьютер, в слот PCI или ISA). Внутренние модемы дешевле, однако внешние модемы предпочтительнее (во внутренних Win-модемах часть функций реализовано не аппаратно, а программно, что увеличивает загрузку процессора компьютера и создает проблемы при настройке на использование с операционными системами, отличными от Windows). В зависимости от области применения, выделяют обычные и профессиональные модемы. Профессиональные модемы более дороги, однако обладают

большей надежностью и наличием средств централизованного управления модемом, для интеграции его в модемный пул сети.

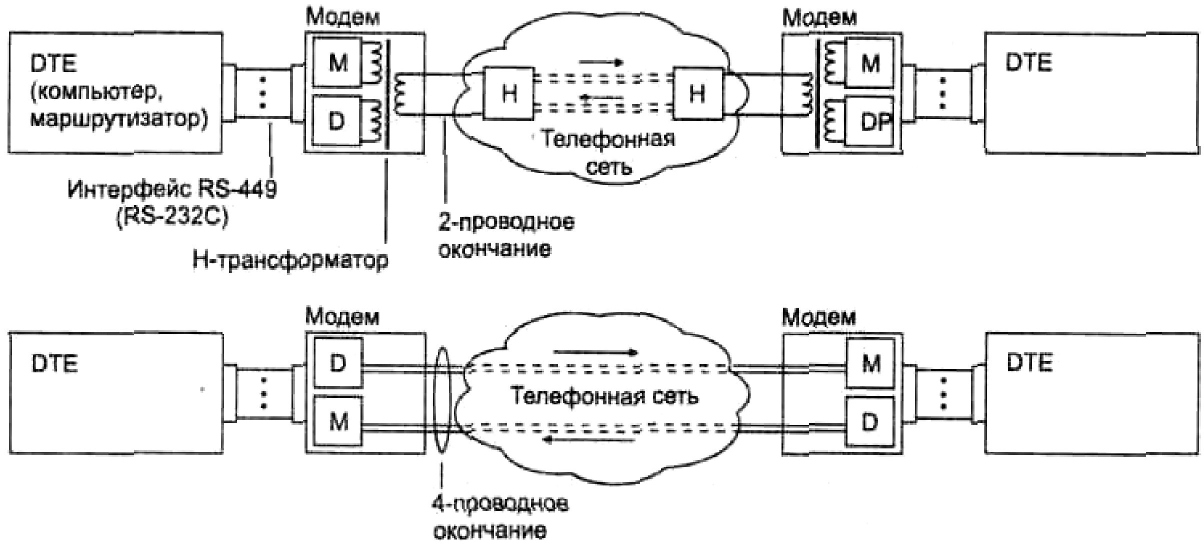


рис. 7.1. Использование модемов на аналоговых телефонных линиях.

Скорость на которой работает модем определяется типом протокола, который поддерживают оба модема, участвующие в соединении. Протоколом также определяются и методы коррекции ошибок. Для модемов, работающим по 2-проводным линиям тональной частоты (обычные телефонные линии) существуют следующие основные протоколы:

- V.21 — дуплексная асинхронная передача на скорости 300 бит/с;
- V.22 — дуплексная асинхронная/синхронная передача на скорости до 1200 бит/с;
- V.22 bis, V.26 ter — скорость передачи до 2400 бит/с;
- V.32 — скорость передачи до 9600 бит/с;
- V.32 bis — скорость передачи 14 400 бит/с;
- V.34 — скорость передачи до 28 800 бит/с;
- V.34+ — скорость передачи до 33 600 бит/с.
- V.42, V.42 bis — этот стандарт развивает ранее использовавшийся в модемах протокол MNP 1-10 фирмы Microcom и содержит процедуры коррекции ошибок, сжатия передаваемых данных (до 1:8), согласования параметров передачи данных, уведомления о приостановке и возобновления передачи данных при асинхронном интерфейсе. В синхронном интерфейсе для коррекции ошибок используется протокол HDLC, а для компрессии – фирменный протокол SDC компании Motorola.
- V.90 — протокол асимметричного обмена данными: со скоростью 56 000 бит/с из сети и со скоростью 33 600 – 40 000 бит/с в сеть. Протокол совместим со стандартом V.34+.

На практике сегодня в основном применяются модемы, работающие по протоколам V34+ и V.90. Стандарт V.34+ позволяет работать по телефонным линиям практически любого качества. Первоначальное соединение модемов происходит по асинхронному интерфейсу на минимальной скорости 300 бит/с, что позволяет работать на самых плохих линиях. После тестирования линии выбираются основные параметры передачи, которые впоследствии могут динамически изменяться без разрыва связи, адаптируясь к изменению качества линии.

Протокол V.90 обеспечивает асимметричный обмен данными: со скоростью 56 Кбит/с из сети и со скоростью 30-40 Кбит/с в сеть. Это значительно повышает скорость доступа к Internet (или другой сети), т.к. объем информации (графика, видео и т.д.), полученные из сети значительно превышает объем информации направленной в сеть (в основном – это запросы на получение web-страниц и файлов). Основная идея протокола V.90 состоит в следующем: при наличии цифровой АТС, единственным аналоговым звеном в соединении с провайдером является телефонная пара, связывающая аналоговый модем компьютера с коммутатором телефонной станции. При прохождении сигнала от модема к АТС, аналоговый сигнал преобразуется в цифровую форму. Это преобразование вносит дополнительные погрешностью дискретизации, что, в сочетании с шумами линии, ограничивает скорость передачи не более 33,6 - 40 Кбит/с. Однако, при прохождении сигнала от цифровой АТС к модему, обратное цифро-аналоговое преобразование не вносит дополнительного шума, что делает возможным увеличение скорости передачи до 56 Кбит/с. Такая скорость может быть достигнута только если и Internet-провайдер подключен к АТС по цифровому интерфейсу. В противном случае, даже при использовании протокола V.90 и у клиента и у провайдера, аналогово-цифровые преобразования на цифровых АТС каждой из сторон ограничат скорость передачи данных в обоих направлениях величиной 33,6 Кбит/с.

Модемы работающие на коммутируемых аналоговых телефонных линиях должны поддерживать процедуру вызова абонента (набор номера в импульсном или тональном режиме, определение занятости линии и т.п.). Для управления модемом по синхронному интерфейсу используется протокол V.25 и V.25 bis. При использовании асинхронного интерфейса, управление модемом, совместимом с стандартом компании Hayes, осуществляется при помощи специальных команд, которые можно отдавать модему даже вручную. Ниже приведены примеры некоторых команд (более подробную информацию можно найти в инструкциях, прилагаемых к модему).

Таблица 7.1.

Некоторые команды HEYS-совместимых модемов

ATZ	Сбрасывает все установки модема в состояние, как если бы модем был только что включен. Здесь AT=attention (внимание) - общий префикс для большинства команд.
AT D P	Снять трубку и набрать номер в импульсном режиме набора. Здесь D=dial (набрать), P=pulse (импульсный режим набора). Пример: команда ATDP5557012 набирает номер 555-70-12 в импульсном режиме набора.
AT D T	Снять трубку и набрать номер в тональном режиме набора. Здесь D=dial (набрать), T=tone (тональный режим набора). Пример 1: команда ATDT5557012 набирает номер 555-70-12 в импульсном режиме набора. Пример 2: команда ATDP8W10T04411P557012 в импульсном режиме набирает 8, для выхода на междугороднюю линию, ждет (W=wait) появления сигнала в линии, набирает 10 для выхода на международную линию, переходит в тональный режим набора (T=tone), набирает номер 04411, затем переходит в импульсный режим набора и набирает номер 555-70-12.
AT A	Переключится в режим приема данных. Здесь A=answer (ответить). Команда используется для ручного ответа на входящий звонок другого модема. Для перевода модема в режим автоматического ответа на входящие звонки, необходимо в строчке инициализации модема указать команду AT S0=2 (отвечать на входящие звонки после 2-го звонка). Команду AT A можно также использовать в случае если при обычном телефонном разговоре понадобилось переключиться на прием-передачу данных. Для этого принимающая сторона выдает модему команду AT A, а передающая сторона выдает своему модему команду AT X1 D.
AT H	положить трубку. Здесь H = hang up (повесить трубку).
AT H1	снять трубку. Команды ATDP и ATDT сами снимают трубку и не требуют этой команды.
AT M0	выключить динамик модема. Возможны также варианты ATM1 (динамик выключен при приеме несущей), ATM2 (динамик всегда включен), ATM3 (динамик выключен при наборе номера и приема несущей).
AT L0	минимальная громкость динамика модема. Возможны также варианты (по степени увеличения громкости): ATL1, ATL2, ATL3.
+ + +	Прекращается передача данных и модем переходит в режим приема AT-команд. Команда работает только при установленном соединении с другим модемом. Символы + + + могут быть заменены на другие, путем записи ASCI-кода соответствующего символа (от 0 до 127) в регистр S2 модема. Пример: S2=33 меняет escape-последовательность на !!!.

В ответ на отдаваемые команды, модем выдает стандартные ответы, например: OK, CONNECT (соединение установлено), BUSY (линия занята), RING (вам звонят), NO CARRIER (пропала несущая = потеряна связь с другим модемом), NO DIALTONE (отсутствует сигнал в телефонной линии), ERROR (ошибка в AT-команде) и др.

7.1.2. Цифровые выделенные линии PDH и SONET/SDH

Цифровая аппаратура PDH была разработана в конце 60-х годов компанией AT&T для решения проблемы связи крупных коммутаторов телефонных сетей между собой. К этому времени аналоговая аппаратура исчерпала свои возможности по пропускной способности, и требовалась либо прокладка новых кабелей большой протяженности, либо изменение принципов работы оборудования. Внедрение цифровой аппаратуры PDH позволило повысить скорость передачи и снизить уровень помех при передаче голоса. Существуют два поколения технологий цифровых первичных сетей:

1) Технология PDH — Plesiochronic Digital Hierarchy, плезиохронная цифровая иерархия ("плезио" означает "почти").
2) Технология SDH — Synchronous Digital Hierarchy, синхронная цифровая иерархия. В Америке технологии SDH соответствует стандарт SONET.

Технология PDH

Первым уровнем скоростей технологии является аппаратура T1, которая позволяет передавать голос и данные со скоростью 1,544 Мбит/с. Первоначально, аппаратура T1 разрабатывалась для передачи по одному каналу голоса 24 абонентов в цифровой форме. Так как абоненты по-прежнему пользуются обычными аналоговыми телефонными аппаратами, то мультиплексор T1 на телефонной станции сам осуществляет

оцифровывание голоса. В результате каждый абонентский канал образовывает цифровой поток данных 64 Кбит/с. Данные 24-х абонентов собираются в кадр достаточно простого формата: в каждом кадре последовательно передается по одному байту каждого абонента, а после 24-х байт вставляется один бит синхронизации. Таким образом, мультиплексор T1 обеспечивает передачу голосовых данных со скоростью 1,544 Мбит/с (24 абонента * 64 Кбит/с + биты синхронизации). Однако при помощи оборудования T1 можно передавать не только голос, но и данные. Для этого компьютер или маршрутизатор должны быть подключены к цифровой выделенной линии при помощи специального устройства DSU/CSU, которое может быть выполнено в отдельном корпусе, или встроено в маршрутизатор. Устройство формирует кадры канала T1,

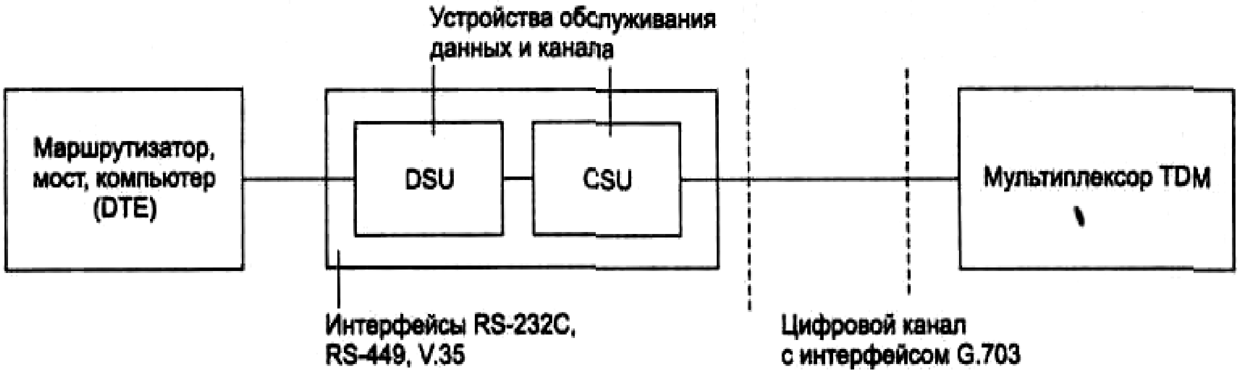


Рис. 7.2. Использование DSU/CSU для подключения к цифровой выделенной линии

Пользователь может арендовать не весь канал T1 (1,544 Мбит/с), а только его часть - несколько каналов 64 Кбит/с. Такой канал называется "дробным" (fractional) каналом T1. Так, например, если пользователь арендовал 3 канала 64 Кбит/с (т.е. канал 192 Кбит/с), то в каждом кадре T1 пользователю будет отведено только 3 байта. Если пользователю необходимо получить скорость выше 1,544 Мбит/с, то для этого необходимо арендовать канал T2 или T3. Четыре канала типа T1 объединяются в канал T2, а семь каналов T2 объединяются в канал T3. Такая иерархия скоростей применяется в США. В Европе используются международные стандарты иерархии скоростей, отличающиеся от стандартов США, и соответствующая аппаратура называется E1, E2, E3. Ниже приведена таблица, иллюстрирующая различия американского и европейского вариантов.

Таблица 7.2.

Иерархия скоростей PDH

Америка				Европа (стандарт CCITT)			
Обору-дование	Количество голосовых каналов	Количество каналов предыдущего уровня	Скорость, Мбит/с	Обору-дование	Количество голосовых каналов	Количество каналов предыдущего уровня	Скорость, Мбит/с
	1	1	64 Кбит/с		1	1	64 Кбит/с
T1	24	24	1,544	E1	30	30	2,048
T2	96	4	6,312	E2	120	4	8,488
T3	672	7	44,736	E3	480	4	34,368
T4*	4032	6	274,176	E4*	1920	4	139,264

* Скорости, соответствующие оборудованию T4 / E4, определены в стандартах, но на практике не используются.

Физический уровень технологии PDH поддерживает различные виды кабелей: витую пару, коаксиальный кабель и волоконно-оптический кабель. Основным вариантом абонентского доступа к каналам T1/E1 является кабель из двух витых пар с разъемами RJ-48. Две пары требуются для организации дуплексного режима передачи данных. Коаксиальный кабель благодаря своей широкой полосе пропускания поддерживает канал T2/E2 или 4 канала T1/E1. Для работы каналов T3/E3 обычно используется либо коаксиальный кабель, либо волоконно-оптический кабель, либо каналы СВЧ. Цифровое абонентское окончание технологии PDH, получило название HDSL (High speed DSL).

Как американский, так и международный варианты технологии PDH обладают несколькими недостатками. Чересчур простой формат кадра PDH, где положение данных канала жестко фиксировано (первый байт – первый канал, второй байт – второй канал и т.д.) приводит к нерациональному использованию кадра. Так если из 24 каналов данные передаются только по одному каналу, то мультиплексор T1 все равно не может передать больше, чем 1 байт данных канала в каждом кадре. Остальные 23 байта кадра просто заполняются нулями. Более того, для того, чтобы выделить из кадра данные только одного канала, придется полностью "разобрать" (демультиплексировать) весь кадр. Другим существенным недостатком технологии

PDH является отсутствие развитых встроенных процедур контроля и управления сетью. Третий недостаток состоит в слишком низких, по современным понятиям, скоростях иерархии PDH. Волоконно-оптические кабели позволяют передавать данные со скоростями в несколько гигабит в секунду по одному волокну, но это свойство технология PDH не реализует — ее иерархия скоростей заканчивается уровнем 139 Мбит/с. Все эти недостатки устранены в новой технологии первичных цифровых сетей, получившей название синхронной цифровой иерархии — Synchronous Distal Hierarchy, SDH.

Технология SONET/SDH

Технология SONET/SDH продолжает иерархию скоростей технологии PDH и позволяет организовать передачу данных со скоростями от 155,520 Мбит/с до 2,488 Гбит/с по оптоволоконному кабелю. Технология синхронной цифровой иерархии первоначально была разработана компанией Bellcore под названием "Синхронные оптические сети" — Synchronous Optical NETs, SONET в 1984 году. Затем эта технология была стандартизована комитетом T1 ANSI и получила название Synchronous Digital Hierarchy, SDH. В терминологии и начальной скорости технологии SDH и SONET остались расхождения, но это не мешает совместимости аппаратуре разных производителей, а технология SONET/ SDH фактически стала считаться единой технологией. В стандарте SDH все уровни скоростей имеют общее название: STM-n — Synchronous Transport Module level n. В технологии SONET существуют два обозначения для уровней скоростей: STS-n — Synchronous Transport Signal level n, употребляемое при передаче данных электрическим сигналом, и OC-n — Optical Carrier level n, употребляемое при передаче данных световым лучом по волоконно-оптическому кабелю. Иерархия скоростей SONET/SDH, представлена ниже.

Таблица 7.3.

Иерархия скоростей SONET/SDH

SDH	SONET	Скорость
—	STS-1, OC-1	51,840 Мбит/с
STM-1	STS-3, OC-3	155,520 Мбит/с
STM-3	STS-9, OC-9	466,560 Мбит/с
STM-4	STS-12, OC-12	622,080 Мбит/с
STM-6	STS-18, OC-18	933,120 Мбит/с
STM-8	STS-24, OC-24	1,244 Гбит/с
STM-12	STS-36, OC-36	1,866 Гбит/с
STM-16	STS-48, OC-48	2,488 Гбит/с

Как видно из таблицы, уровень STM-1 технологии SDH (155,520 Мбит/с) может переносить кадры уровня E4 технологии PDH (139,264 Мбит/с). Таким образом достигается преемственность технологий PDH и SDH. Помимо более высокой скорости передачи данных, технология SDH имеет и другие преимущества. Кадр SDH имеет заголовок достаточно сложного формата, благодаря которому данные каждого канала пользователя жестко не привязаны к своему положению в кадре. Данные канала пользователя укладываются в так называемый "виртуальный контейнер" – своего рода подкадр, изолирующий данные одного канала пользователя от другого. Виртуальный контейнер может быть смещен относительно начала поля данных кадра SDH на произвольную величину или даже находится в различных смежных кадрах SDH. Технология SDH сама подбирает виртуальные контейнеры подходящего формата для различных каналов пользователя, следит за тем, чтобы наиболее рационально уложить в кадр "мозаику" из виртуальных контейнеров, а также позволяет объединять виртуальные контейнеры в контейнеры более высокого уровня. Техника виртуальных контейнеров позволяет извлекать (добавлять) отдельные пользовательские каналы из кадра SDH, не производя его полное демультиплексирование ("разборку").

К другим преимуществам технологии SDH относится высокая отказоустойчивость, которая в сети SONET/SDH встроена в ее основные протоколы. Этот механизм называется автоматическим защитным переключением — Automatic Protection Switching, APS. Существуют два способа его работы. В первом способе защита осуществляется по схеме 1:1. Для каждого рабочего волокна (и обслуживающего его порта) назначается резервное волокно. Во втором способе, называемом 1:n, для защиты n волокон назначается только одно защитное волокно. Обычно при защите 1:1 используется схема двух колец, похожая на двойные кольца FDDI (см. лекции ранее), хотя может использоваться и обычная схема подключения типа "звезда".

Управление, конфигурирование и администрирование сети SONET/SDH также встроено в протоколы. Служебная информация протокола позволяет централизованно и дистанционно конфигурировать пути между конечными пользователями сети, изменять режим коммутации потоков, а также собирать подробную статистику о работе сети. Существуют мощные системы управления сетями SDH, позволяющие прокладывать новые каналы простым перемещением мыши по графической схеме сети.

Технологии PDH и SDH широко используются для построения корпоративных сетей. На основе выделенных линий SDH можно строить сети с коммутацией пакетов, например Frame Relay или ATM, или же сети с коммутацией каналов, например ISDN. Технология ATM облегчила эту задачу, приняв стандарты SDH в качестве основных стандартов физического уровня (подробнее о сетях ATM см. далее в лекциях).

7.1.3. Цифровые коммутируемые линии ISDN (сети ISDN)

Сети ISDN (Integrated Services Digital Network, цифровые сети с интегральными услугами) задумывались как цифровые сети, идущие на смену телефонным сетям, и обеспечивающие качественную и быструю передачу голоса и компьютерных данных. Сети ISDN предоставляют много услуг, среди которых выделенные и коммутируемые цифровые каналы передачи данных и голоса, сеть передачи данных с коммутацией пакетов (аналогично сети X.25), услуги сети Frame Relay. В настоящее время большинство из услуг сети ISDN не востребованы: сети Frame Relay выделились в самостоятельные сети, сети с коммутацией пакетов в рамках ISDN не обеспечивают высокой скорости и гарантий качества обслуживания, как это делают сети ATM. Поэтому в основном сети ISDN используются так же, как и аналоговые телефонные сети, но только как более скоростные и надежные.

Если технологии PDH и SDH/SONET используются для создания выделенных цифровых линий, то технология ISDN, помимо этого, позволяет организовать ***коммутируемую*** цифровую линию, со скоростью передачи данных до 128 Кбит/с (2-проводное окончание), либо до 1,544 или 2,048 Мбит/с (4-проводное окончание, скорость линий T1/E1). Абонент сети ISDN может установить соединение с другим абонентом, при помощи адреса ISDN, который напоминает телефонный номер. Адрес ISDN состоит из двух частей: номера абонента (до 15 десятичных цифр: код страны, код города, номер абонента, который соответствует точке подключения сети абонента к сети ISDN) и адреса абонента (подномер конкретного устройства в сети абонента). Например, если на предприятии имеется офисная АТС, подключенная к сети ISDN, то ей можно присвоить номер абонента 17-095-6402000, а для конкретного терминального устройства (компьютер, телефон) внутри предприятия выделить внутренний номер 134. Тогда для связи с этим устройством придется набрать номер 17-095-6402000-134. Если сеть ISDN используется для доступа к другой сети (например, X.25 или TCP/IP), то номер абонента останется прежним 17-095-6402000, а в поле "адрес абонента" будут указаны не цифры 134, а адрес компьютера в том формате, который принят в подключаемой сети. Для этого перед адресом абонента указывается специальный префикс, свидетельствующий о формате адреса.

Для использования услуг сети ISDN в помещении пользователя должно быть установлено специальное оборудование CPE (Customer Premises Equipment), которое состоит из сетевого окончания NT (Network Termination) и терминального оборудования TE (Terminal Equipment). Именно сетевое окончание NT получает номер абонента, а терминальное оборудование TE (компьютер, маршрутизатор или телефонный аппарат с интерфейсом ISDN) получает адрес абонента. Примечание: в Европе устройство NT принято считать частью оборудования сети и выпускать отдельно, в США устройство NT считается частью пользовательского оборудования и встраивается в маршрутизатор.

Как уже упоминалось выше, сеть ISDN позволяет организовывать коммутируемые соединения со скоростью до 1,544 (2,048) Мбит/с. Однако пользователь имеет возможность оплачивать и канал с гораздо меньшей пропускной способностью. Достигается это путем комбинирования каналов трех типов:

- B — со скоростью передачи данных 64 Кбит/с;
- D — со скоростью передачи данных 16 или 64 Кбит/с;
- H — со скоростью передачи данных 384 Кбит/с (H0), 1536 Кбит/с (H11) или 1920 Кбит/с (H12).

Каналы типа B обеспечивают передачу пользовательских данных (оцифрованного голоса, компьютерных данных или смеси голоса и данных) со скоростью 64 Кбит/с, или с более низкими скоростями, если пользовательское оборудование разделит канал B на подканалы (сеть ISDN всегда коммутирует только целые каналы типа B). Канал типа B может также подключать пользователя к коммутатору сети X.25.

Канал типа D выполняет две основные функции. Первой и основной является передача адресной информации для того, чтобы коммутаторы ISDN могли установить соединение между двумя пользователями (коммутация каналов типа B). Второй функцией канала D (если он не занят) является оказание услуг низкоскоростной сети с коммутацией пакетов, аналогичной сети X.25.

Каналы типа H предоставляют пользователям возможности высокоскоростной передачи данных. На них могут работать службы высокоскоростной передачи факсов, видеоинформации, качественного воспроизведения звука.

Пользовательский интерфейс ISDN (то за что платит пользователь) представляет собой набор из перечисленных выше каналов. Сеть ISDN поддерживает два типа пользовательского интерфейса — начальный (Basic Rate Interface, BRI) и основной (Primary Rate Interface, PRI). Начальный интерфейс BRI подключается к сети по 2-проводному окончанию и работает по схеме 2B+D, или B+D, или просто D. Основной интерфейс PRI подключается к сети по 4-проводному окончанию и предназначен для пользователей с повышенными требованиями к пропускной способности сети. Интерфейс PRI поддерживает либо схему 23B+D (США и Япония), либо схему 30B+D (Европа). Возможны варианты интерфейса PRI с меньшим количеством каналов типа B, например 20B+D. При установке у пользователя нескольких интерфейсов PRI все они могут иметь один канал типа D, при этом получается схема 2*30B+B+D или 2*23B+B+D. Для каналов H возможны интерфейсы 3H0+D или H11+D (США), и 5H0+D или H12+D (Европа). В любом случае, суммарная скорость любого набора каналов по одному интерфейсу PRI не должна превышать скорость линий T1/E1: 1,544 Мбит/с (США) или 2,048 Мбит/с (Европа).

7.1.4. Асимметричные цифровые абонентские линии ADSL

Основной проблемой при подключении удаленного пользователя к сети Internet или к сети предприятия является проблема "последней мили". В самой глобальной сети могут использоваться высокоскоростные линии SONET/SDH, однако пользователь обычно подключается к сети при помощи обычного аналогового модема, что ограничивает скорость обмена данными величиной в 33,6 – 56 Кбит/с, вне зависимости от того, насколько "быстрая" сама сеть. Устанавливать у каждого пользователя дома оборудование T1/E1 или ISDN с 4-проводным окончанием слишком дорого и технически сложно, а оборудование ISDN с 2-проводным окончанием обеспечивает недостаточно высокую скорость доступа (всего 128 Кбит/с). Большинство пользователей хотели бы получить дешевый и быстрый цифровой доступ к Internet (и через Internet к сети своего предприятия) при помощи стандартной 2-проводной телефонной линии и простого устройства типа модема. Перечисленные ниже технологии решают проблему "последней мили" при помощи специальных модемов: симметричная цифровая абонентская линия (SDSL), цифровая абонентская линия с переменной скоростью (Rate Adaptive DSL, RADSL), сверхбыстрая цифровая абонентская линия (Very high-speed DSL, VDSL), асимметричная цифровая абонентская линия (Asymmetric Digital Subscriber Line, ADSL). Среди перечисленных, наибольшее распространение получила технология ADSL.

Технология ADSL рассчитана на высокоскоростную передачу данных на коротком отрезке витой пары, соединяющей абонента с ближайшей телефонной АТС. В то время как обычные аналоговые модемы (V.34+, V.90) рассчитаны на работу через телефонную сеть с произвольным количеством телефонных коммутаторов между клиентом и провайдером, модемы ADSL могут подключаться только непосредственно к маршрутизатору на телефонной станции, не проходя телефонные коммутаторы (благодаря этому на коротком отрезке витой пары между пользователем и маршрутизатором удается добиться высоких скоростей передачи данных). То есть, если телефонная станция предоставляет услуги ADSL, то в здании АТС должен находится маршрутизатор сети Internet (на рисунке обозначен "R"), который при помощи высокоскоростных каналов связан с другими маршрутизаторами. Оборудование ADSL подключается непосредственно к маршрутизатору. Голосовой канал выделяется оборудованием ADSL и направляется на коммутатор телефонной станции. Одним из главных преимуществ технологии ADSL, по сравнению с аналоговыми модемами, ISDN и T1/E1, — это то, что передача голоса и данных идут параллельно и никак не влияют друг на друга.

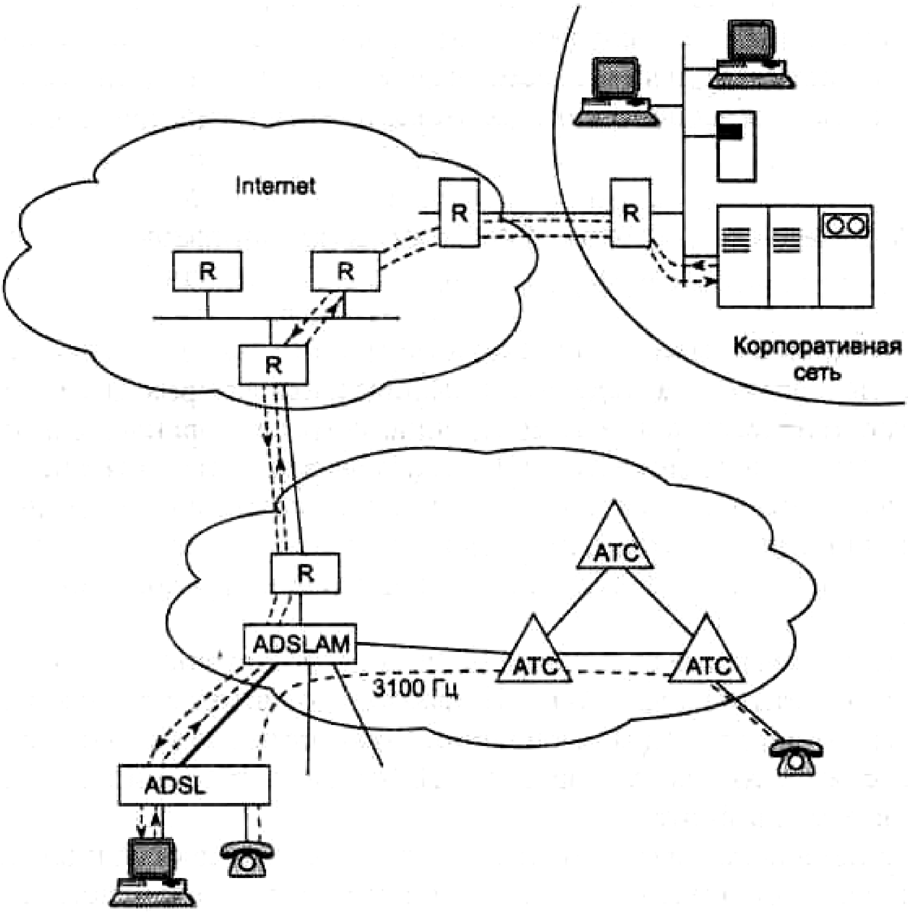


рис. 7.3. Технология ADSL.

ADSL-модемы, подключаемые к обоим концам короткой линии между абонентом и АТС, образуют три канала: быстрый канал передачи данных из сети в компьютер, менее быстрый дуплексный канал передачи данных из компьютера в сеть и простой канал телефонной связи, по которому передаются обычные

телефонные разговоры. Передача данных в канале "сеть-абонент" происходит со скоростью от 1,5 до 6 Мбит/с, в канале "абонент-сеть" — со скоростью от 16 Кбит/с до 1 Мбит/с. В обоих случаях конкретная величина скорости передачи зависит от длины и качества линии. Асимметричный характер скорости передачи данных вводится специально, так как удаленный пользователь Internet или корпоративной сети обычно загружает данные из сети в свой компьютер, а в обратном направлении идет либо подтверждение приема данных, либо поток данных существенно меньшей скорости.

Если центральная сеть предприятия подключена к Internet через выделенный высокоскоростной канал, то все удаленные пользователи, у которых установлены модемы ADSL, получают высокоскоростной доступ к сети своего предприятия на тех же телефонных каналах, которые всегда соединяли их с городской АТС. Кроме абонентских окончаний телефонных сетей в последнее время для скоростного доступа к Internet стали применять абонентские окончания кабельного телевидения. Для этих целей уже разработан специальный вид модемов — кабельные модемы. В кабельных модемах используется имеющийся коаксиальный 75-омный телевизионный кабель для передачи данных из сети в компьютер со скоростью до 30 Мбит/с, а из компьютера в сеть — со скоростью до 10 Мбит/с. При этом качество передаваемых сигналов очень высокое. Высокоскоростные абонентские окончания создают для поставщиков услуг Internet дополнительную проблему — им необходимо иметь очень скоростные каналы доступа к остальной части Internet, так как 10 абонентов с трафиком по 8 Мбит/с создают общий трафик в 80 Мбит/с, который качественно можно передать только с помощью технологий SONET/SDH или ATM.

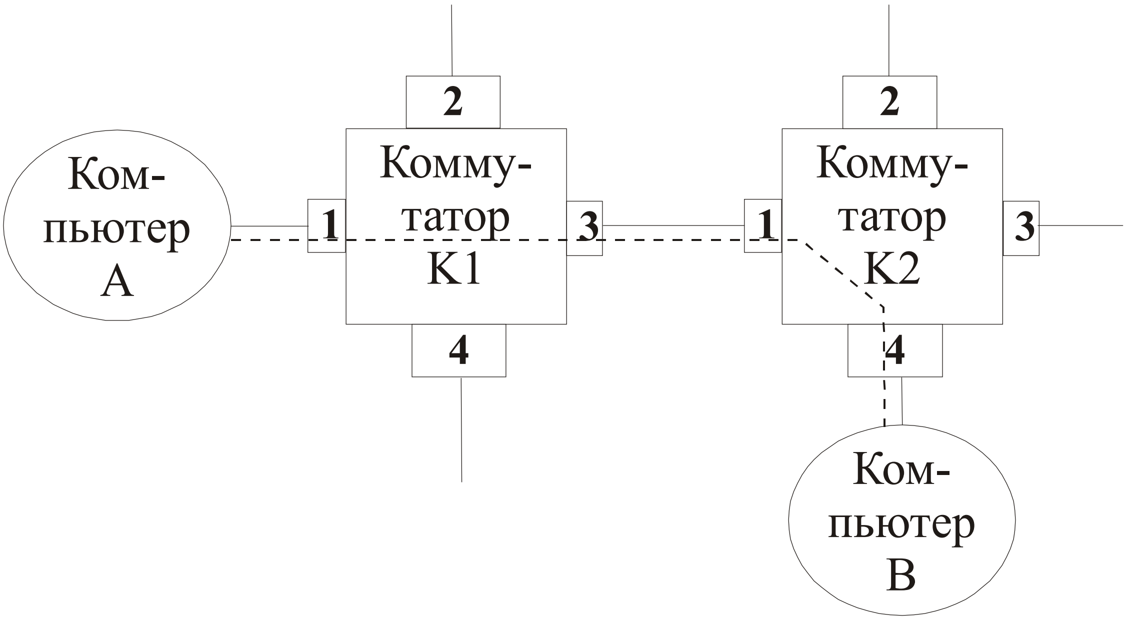
7.2. Глобальные сети с коммутацией пакетов.

Имея выделенный или коммутируемый канал, можно напрямую объединить между собой несколько локальных сетей при помощи удаленных маршрутизаторов или мостов. В самом простом варианте это будет реализовано при помощи компьютера-шлюза, на котором настроен интерфейс вызова по требованию: пакеты циркулируют в рамках локальной сети, а если на шлюз попадает пакет, направляющийся в другую локальную сеть, то модем компьютера-шлюза сам позвонит на другой компьютер-шлюз (телефонный номер выбирается в зависимости от адреса назначения пакета), а после передачи пакета разорвет соединение.

Однако такая схема не всегда экономически оправдана. Предположим, что одна локальная сеть находится в РБ, а вторая в США. Даже при небольшом трафике между сетями международные разговоры обойдутся очень дорого. Поэтому будет разумнее для соединения локальных сетей воспользоваться услугами уже существующих глобальных сетей, например Internet. Такая возможность предоставляется, уже рассматривавшейся ранее, технологией виртуальных частных сетей (VPN – Virtual Private Network), которая благодаря шифрованию позволяет организовать безопасное соединение двух ЛВС через Internet. Однако сеть Internet для этих целей стала использоваться сравнительно недавно, да и отнюдь не является самым быстрым, надежным и безопасным вариантом. Существует большое количество других глобальных сетей с коммутацией пакетов, позволяющих решать эти задачи. К глобальным сетям с коммутацией пакетов относятся сети X.25, Frame Relay, ATM и TCP/IP (Internet). Принципы работы сетей TCP/IP уже были частично рассмотрены ранее, поэтому сейчас дадим краткий обзор остальных глобальных сетей.

Сети X.25, Frame Relay и ATM состоят из коммутаторов (которые отличаются от коммутаторов локальных сетей), объединенных между собой связями "точка-точка" и работающими с установлением виртуального канала между абонентами сети. Под виртуальным каналом понимается нумерованное соединение между двумя абонентами, в котором данные передаются не на основании таблиц маршрутизации коммутаторов, а на основании номера виртуального канала. Точнее говоря маршрутизация пакетов между коммутаторами сети на основании таблиц маршрутизации происходит только один раз — при создании виртуального канала. После создания виртуального канала передача пакетов коммутаторами происходит на основании номера или идентификатора виртуального канала (Virtual Channel Identifier, VCI). Рассмотрим процесс создания виртуального канала.

Допустим компьютер А хочет установить соединение с компьютером В (см. рис. 7.4.). Для этого в коммутатор К1 направляется специальный пакет — запрос на установление соединения (Call Request), который содержит адрес узла назначения. При отправке, пакету Call Request компьютер А назначает номер виртуального канала (VCI), равный 4. Номер VCI выбирается компьютером произвольно, исходя из тех соображений, что от компьютера уже было ранее проложено три виртуальных канала к другим компьютерам. Пакет Call Request поступает на коммутатор К1, тот анализирует свою внутреннюю таблицу маршрутизации и определяет, что пакеты до компьютера В необходимо направлять на порт 3. Кроме того, коммутатор присваивает пакету Call Request новый VCI = 10. Это связано с тем, что через коммутатор уже проложено 9 соединений и старый VCI=4 использоваться не может. Одновременно с изменением номера VCI, коммутатор делает записи в таблицах коммутации портов 1 и 3 (см. рис. 7.4.).



VCI-входящий	VCI-исходящий	Порт
Таблица коммутации порта 1 коммутатора К1		
4	10	3
Таблица коммутации порта 3 коммутатора К1		
10	4	1

VCI-входящий	VCI-исходящий	Порт
Таблица коммутации порта 1 коммутатора К2		
10	25	4
Таблица коммутации порта 4 коммутатора К2		
25	10	1

Рис. 7.4. Коммутация в сетях с виртуальными каналами.

Запись для порта 1 коммутатора К1, приведенная на рисунке, означает: все последующие пакеты, которые поступят на порт 1 со значением VCI=4, необходимо перенаправлять на порт 3 и присваивать им новое значение VCI=10. Запись для порта 3 коммутатора К1, приведенная на рисунке, означает: все пакеты, которые вернутся на порт 3 со значением VCI=10, необходимо направлять на порт 1 и присваивать им старое значение VCI=4. Аналогичная схема повториться и в коммутаторе К2, только в связи с тем, что он еще более загружен (уже проложено 24 соединения), значение VCI пакета опять изменится и станет равно 25. Именно такой пакет (VCI=25) попадет в компьютер В, тот ответит установлением соединения и в дальнейшем будет работать с компьютером А по виртуальному каналу VCI=25. Коммутаторы, используя свои таблицы коммутации, сами произведут все промежуточные преобразования номера VCI и компьютер А получит ответ по виртуальному каналу VCI=4, т.е. по тому каналу, по которому и запрашивалось установление соединения. Таким образом, каждый из компьютеров работает с постоянным номером виртуального канала VCI, хотя по пути следования номер VCI изменяется в каждом коммутаторе.

Виртуальный канал может быть как коммутируемый (Switched Virtual Circuit, SVC), так и постоянный (Permanent Virtual Circuit, PVC). Создание коммутируемого виртуального канала осуществляется автоматически коммутаторами сети, по запросу абонента, а создание постоянного виртуального канала происходит заранее, вручную администратором сети. Таблицы маршрутизации сетей с виртуальными каналами аналогичны таблицам маршрутизации протокола IP, но имеют более простую структуру каждой записи. Запись состоит из адреса назначения и номера порта, на который нужно переслать пакет. Адрес следующего коммутатора не нужен, так как все связи между коммутаторами являются связями типа "точка-точка". Стандарты глобальных сетей обычно не описывают какой-либо протокол обмена маршрутной информацией, подобный RIP или OSPF, позволяющий коммутаторам сети автоматически строить таблицы маршрутизации. Поэтому в таких сетях администратор обычно вручную составляет таблицу маршрутизации, указывая для обеспечения отказоустойчивости основной и резервный пути для каждого адреса назначения. Исключением являются сети ATM, для которых разработан протокол маршрутизации PNNI, аналогичный протоколу OSPF.

Техника виртуальных каналов имеет свои достоинства и недостатки по сравнению с обычной IP-маршрутизацией. Основным преимуществом является ускорение доставки пакетов, т.к. коммутатор должен просматривать не большие таблицы маршрутизации, содержащие информацию о всей сети, а только небольшие таблицы коммутации портов. Кроме того, вместе с каждым пакетом передается не полный адрес компьютера-назначения, а только небольшой номер виртуального канала, что снижает долю служебной информации в пакете, увеличивая тем самым скорость передачи пользовательских данных. Однако за уменьшение служебного заголовка пакета приходится платить невозможностью распараллеливания трафика, а при отказе какого-либо канала соединение приходится также устанавливать заново.

Маршрутизация каждого пакета без предварительного установления соединения (IP-маршрутизация) эффективна для кратковременных потоков данных. При использовании виртуальных каналов очень эффективно передаются долговременные потоки. В связи с этим, компания Ipsilon разработала технологию IP-switching, которая вводит в сети ATM возможность работать как по виртуальным каналам, так и без предварительного установления соединения. Эта технология стала достаточно популярной, хотя и не приобрела статус стандарта.

7.2.1. Сети X.25

Сети X.25 являются самыми первыми сетями с коммутацией пакетов, использованными для объединения корпоративных сетей. Первоначально сети разрабатывались для низкоскоростной передачи данных по линиям связи с большим уровнем помех, и использовались для подключения банкоматов, кассовых терминалов, принимающих кредитные карточки, и для соединения сетей предприятий между собой. Долгое время сеть X.25 была единственной широко распространенной коммерческой сетью (сеть Internet, как коммерческая стала эксплуатироваться совсем недавно), поэтому для корпоративных пользователей выбора не было. В настоящее время сеть X.25 продолжает успешно эксплуатироваться, используя высокоскоростные цифровые линии связи для соединения своих коммутаторов. Так, в частности, большинство банков и промышленных предприятий запада используют сеть X.25 для организации удаленного доступа к своим сетям.

Сеть X.25 состоит из коммутаторов, соединенных между собой по схеме "точка-точка", и работающих с установлением виртуального канала. Для связи коммутаторов могут использоваться цифровые линии PDH / SDH или аналоговые модемы, работающие по выделенной линии. Компьютеры (маршрутизаторы), поддерживающие интерфейс X.25, могут подключаться к коммутатору непосредственно, а менее интеллектуальные терминалы (банкоматы, кассовые аппараты) – при помощи специального устройства PAD (Packet Assembler Disassembler). PAD может быть встроенным в коммутатор или удаленным. Терминалы получают доступ ко встроенному PAD по телефонной сети с помощью модемов (встроенный PAD также подключается к телефонной сети с помощью нескольких модемов). Удаленный PAD представляет собой небольшое автономное устройство, находящееся в помещении клиента и подключенное к коммутатору через выделенную линию. К удаленному PAD терминалы подключаются через COM-порт (интерфейс RS-232C). Один PAD обычно обеспечивает доступ для 8, 16 или 24 терминалов. Терминалы не имеют конечных адресов в сети X.25 - адрес присваивается только порту PAD.

Адресация в сетях X.25 строится по следующему принципу: в адресе используются десятичные цифры, длина адреса не может превышать 16 цифр. Если сеть X.25 не связана с внешним миром, то она может использовать любой адрес. Если же сеть X.25 планирует связываться с другими сетями, то необходимо придерживаться международного стандарта адресации (стандарт X.121 - International Data Numbers, IDN).

<table>
<tr><td colspan="2">4 цифры - код идентификации сети
(Data Network Identification Code, DNIC). Из них:</td><td rowspan="2">Остальные цифры - номер национального терминала (National Terminal Number, NTN).

Соответствуют адресу компьютера в сети.</td></tr>
<tr><td>3 цифры - определяют страну, в которой находится сеть X.25</td><td>1 цифра - номер сети X.25 в данной стране.</td></tr>
</table>

рис. 7.5. Формат адреса в сети X.25

Из приведенного на рис. 7.5 формата адреса видно, что в одной стране может быть только 10 сетей X.25. Если же требуется пронумеровать больше, чем 10 сетей, то одной стране дается несколько кодов. Например, Россия имела до 1995 года один код — 250, а в 1995 году ей был выделен еще один код — 251.

В адресе могут использоваться не только цифры, но и произвольные символы (для этого к адресу надо добавить специальный префикс), что позволяет универсальным коммутаторам, например коммутаторам сети ISDN, работать с пакетами сети X.25.

Основным недостатком сети X.25 является то, что она не дает гарантий пропускной способности сети. Максимум на что она способна – это устанавливать приоритеты для отдельных виртуальных каналов. Поэтому сеть X.25 используется только для передачи компьютерных данных с небольшой пульсацией трафика, и не пригодна для передачи трафика, чувствительного к задержкам (например, голоса). Решением этой проблемы занимаются сети Frame Relay и ATM.

7.2.2. Сети Frame Relay.

Сети Frame Relay разрабатывались как общественные сети для соединения частных локальных сетей и обеспечивают скорость доступа к сети до 2 Мбит/с. Сети Frame Relay, как и сети X.25, состоят из коммутаторов, соединенных между собой по схеме "точка-точка", и работающих с установлением виртуального канала. Услуги Frame Relay обычно предоставляют те же операторы, которые эксплуатируют

сети X.25. Большая часть выпускаемых коммутаторов могут работать, как по протоколам X.25, так и по протоколам Frame Relay.

Отличием сетей Frame Relay от сетей X.25 является то, что если в сети X.25 определен собственный протокол сетевого уровня, то сети Frame Relay работают только на физическом и канальном уровнях, что делает удобной передачу других сетевых протоколов (например TCP/IP или NetBIOS) через сети Frame Relay. Фактически сети Frame Relay начинают занимать в глобальных сетях то же место, что и протокол Ethernet в локальных сетях.

Другой важной особенностью технологии Frame Relay является то, что она позволяет гарантировать качество обслуживания (Quality of Service, QoS). Пользователь всегда оплачивает только ту пропускную способность сети, которая ему действительно нужна, и всегда имеет гарантию, что в любой момент времени эта пропускная способность будет ему предоставлена. Для каждого виртуального канала заключается контракт на качество обслуживания, который включает в себя несколько параметров:

1) CIR (Committed Information Rate) — согласованная информационная скорость, с которой сеть будет передавать данные пользователя. Сеть гарантирует доставку кадров при этой скорости. В общем случае пользователь должен передать в сеть данные со средней скоростью, не превосходящей CIR.
2) Bc (Committed Burst Size) — согласованный объем пульсации. Под пульсацией понимается кратковременное увеличение скорости передачи данных. То есть параметр Bc оговаривает максимальное количество байтов, которое сеть будет передавать от этого пользователя за интервал времени T (T = Bc/CIR). Если пользователь превышает порог Bc, то сеть помечает такой кадр признаком DE=1 (Discard Eligibility), то есть как кадр, подлежащий удалению. Однако кадры, отмеченные DE=1, удаляются из сети только в том случае, если коммутаторы сети испытывают перегрузки. Если же перегрузок нет, то кадры с признаком DE=1 доставляются адресату.
3) Be (Excess Burst Size) — дополнительный объем пульсации, то есть максимальное количество байтов, которое сеть будет пытаться передать сверх установленного значения Bc за интервал времени T. Если пользователь превышает величину Bc+Be, то кадр отбрасывается немедленно.

Пользователь платит именно за величины CIR, Bc и Be. Параметры виртуального канала в направлении от компьютера A к компьютеру B, могут отличаться от параметров в обратном направлении, т.к. компьютеры A и B – это разные пользователи, каждый из которых по своему оплачивает услуги сети Frame Relay. Пользователь может договориться о включении не всех параметров качества обслуживания в контракт. Например, можно использовать только параметры CIR и Bc. Этот вариант дает более качественное обслуживание, так как кадры никогда не отбрасываются коммутатором сразу. Коммутатор только помечает кадры, которые превышают порог Bc за время T, признаком DE=1. Если сеть не сталкивается с перегрузками, то кадры такого канала всегда доходят до конечного узла, даже если пользователь постоянно нарушает договор с сетью. Популярен еще один вид заказа на качество обслуживания, при котором оговаривается только порог Be, а скорость CIR полагается равной нулю. Все кадры такого канала сразу же отмечаются признаком DE=1, но отправляются в сеть, а при превышении порога Be они отбрасываются. Контрольный интервал времени T в этом случае вычисляется как Be/R, где R — скорость доступа канала.

Контракты на качество обслуживания должны заключаться таким образом, чтобы сумма средних скоростей виртуальных каналов не превосходила возможностей портов коммутаторов. При заказе постоянных каналов за это отвечает администратор, а при установлении коммутируемых виртуальных каналов — программное обеспечение коммутаторов. Контракт заключается автоматически, по протоколу Q.931, следующим образом:

1) Абонент сети frame relay, который хочет установить коммутируемое виртуальное соединение с другим абонентом, должен передать в сеть сообщение SETUP, которое имеет несколько параметров, в том числе адрес назначения и запрашиваемые величины CIR, Bc и Be для двух направлений.
2) Коммутатор, с которым соединен пользователь, сразу же передает пользователю пакет CALL PROCEEDING (вызов обрабатывается). Затем он анализирует параметры, указанные в пакете, и если коммутатор может их удовлетворить, то пересылает сообщение SETUP следующему коммутатору.
3) Если все коммутаторы на пути к конечному узлу согласны принять запрос, то пакет SETUP передается в конечном счете вызываемому абоненту. Вызываемый абонент немедленно передает в сеть пакет CALL PROCEEDING и начинает обрабатывать запрос. Если запрос принимается, то вызываемый абонент передает в сеть новый пакет — CONNECT (соединение), который проходит в обратном порядке по виртуальному пути. Все коммутаторы должны отметить, что данный виртуальный канал принят вызываемым абонентом.
4) При поступлении сообщения CONNECT вызывающему абоненту, он должен передать в сеть пакет CONNECT ACKNOWLEDGE (соединение подтверждено). Сеть по цепочке передает вызываемому абоненту пакет CONNECT ACKNOWLEDGE, и на этом соединение считается установленным. По виртуальному каналу могут передаваться данные.

При правильно взятых на себя обязательствах по качеству обслуживания, в сети Frame Relay не возникают перегрузки, а кадры "нарушители", помеченные признаком DE=1 просто отбрасываются. Однако отбрасывание кадров – не единственный способ управления пропускной способностью сети. Существует механизм

оповещения конечных пользователей о том, что в коммутаторах сети возникли перегрузки. Бит FECN (Forward Explicit Congestion Bit) кадра извещает о перегрузках принимающую сторону, которая должна с помощью протоколов более высоких уровней (TCP/IP, SPX и т. п.) известить передающую сторону о том, что та должна снизить интенсивность отправки пакетов в сеть. Бит BECN (Backward Explicit Congestion Bit) напрямую извещает о перегрузках в сети передающую сторону и является рекомендацией немедленно снизить темп передачи.

Сети Frame Relay продолжают оставаться достаточно популярной услугой, особенно для объединения сетей предприятий, однако они не позволяют качественно передавать видеоизображение, да и качественная передача голоса возможна только при условии низкой загрузки сети. Связано это с низкой скоростью доступа к сети (2 Мбит/с), большим размером кадра сети Frame Relay и недостаточно продуманным набором параметров качества обслуживания (так, например, отсутствуют гарантии на задержку передачи кадра и т.п.). Всех этих недостатков лишена технология ATM.

7.2.3. Сети ATM

Сеть ATM работает с установлением виртуального канала и позволяет качественно передавать компьютерные данные, видеоизображение и голос со скоростью от 155 до 622 Мбит/с (сети ATM используют на физическом уровне технологию SONET/SDH, принимая ее иерархию скоростей). Сеть ATM имеет структуру, сходную со структурой сетей X.25 и Frame Relay: конечные станции соединяются каналами "точка-точка" с коммутаторами нижнего уровня, которые в свою очередь соединяются с коммутаторами более высоких уровней. Конечные узлы в сети ATM имеют адреса длиной в 20 байт из которых: 1 байт определяет один из возможных форматов адреса (Authority and Format Identifier, AFI), 8 байт – основная часть адреса (до 15 цифр: код страны, код города, номер абонента – аналогично номеру абонента в сети ISDN), 4 байта номер сети/подсети ATM, 6 байт номер конечного узла в сети ATM (MAC-адрес сетевой карты компьютера), 1 байт – поле селектора (вспомогательное поле). Таблицы маршрутизации коммутаторов составляются вручную, или при помощи протокола PNNI. Установленные виртуальные каналы (выделенные или коммутируемые) нумеруются при помощи идентификатора виртуального канала (Virtual Channel Identifier, VCI). Несколько виртуальных каналов, проходящих через одни и те же коммутаторы, могут объединяться в один виртуальный путь (Virtual Path Identifier, VPI). Так как виртуальных путей меньше, чем виртуальных каналов, то и записей в таблице коммутации портов будет меньше, что ускоряет коммутацию.

Важной отличительной чертой сети ATM является маленький размер пакета данных (53 байта) и хорошо проработанная система параметров качества обслуживания (QoS), что позволяет в равной степени хорошо передавать по сети, как компьютерный трафик (объединение локальных сетей), так и мультимедийный трафик (видеоизображение, голос).

Трафик вычислительных сетей имеет ярко выраженный асинхронный и пульсирующий характер. Компьютер посылает пакеты в сеть в случайные моменты времени, по мере возникновения в этом необходимости. При этом интенсивность посылки пакетов в сеть и их размер могут изменяться в широких пределах. Чувствительность компьютерного трафика к потерям данных высокая, так как без утраченных данных обойтись нельзя, и их необходимо восстановить за счет повторной передачи. В то же время, чувствительность компьютерного трафика к задержкам передачи пакетов данных незначительна. Мультимедийный трафик (голос, видео) характеризуется низким коэффициентом пульсаций, высокой чувствительностью к задержкам передачи данных (отражающихся на качестве воспроизводимого сигнала) и низкой чувствительностью к потерям данных (из-за инерционности физических процессов потерю отдельных замеров голоса или кадров изображения можно компенсировать сглаживанием на основе предыдущих и последующих значений). Сложность совмещения компьютерного и мультимедийного трафика в одной сети проиллюстрирована на рис. 7.6.

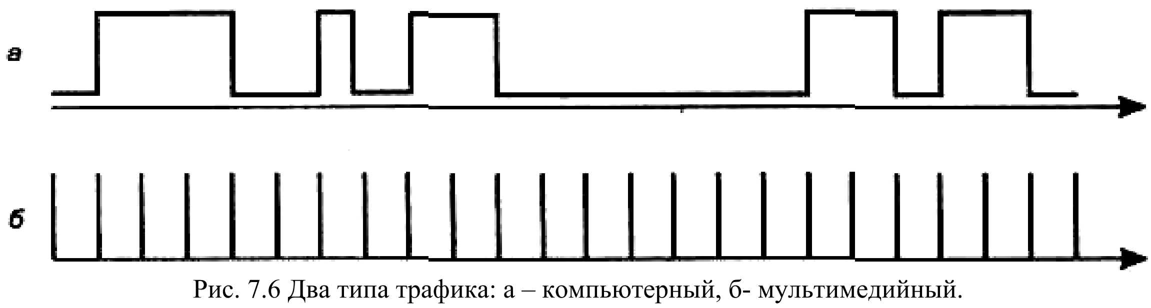


Рис. 7.6 Два типа трафика: а – компьютерный, б- мультимедийный.

На возможности совмещения этих двух видов трафика большое влияние оказывает размер компьютерных пакетов. Если в сети допускаются большие размеры пакетов данных, то один единственный большой пакет может "занять" порт коммутатора и затормозить передачу всех остальных пакетов, что не допустимо для мультимедийного трафика. Кроме того, если минимально допустимый размер пакета велик, то в одном пакете данных (чтобы не терять впустую место) будет передаваться несколько замеров голоса. В результате, первый замер голоса, помещаемый в пакет, будет отправлен не сразу же, а только после того, как в пакет

будут помещены все остальные замеры, что приведет к значительным задержкам и потере качества передачи голоса. Поэтому в сети ATM данные передаются в небольших ячейках (пакетах) фиксированного размера (53 байта: поле данных – 48 байт, заголовок — 5 байт).

Однако использование небольших ячеек фиксированного размера еще не решает всей проблемы. Для полного решения задачи равноправного совмещения в одной сети компьютерного и мультимедийного трафика, технология ATM использует хорошо разработанную схему заказа пропускной способности и качества обслуживания. Соглашение между программой, передающей данные в сеть, и сетью ATM называется трафик-контрактом. Основным его отличием от соглашений, применяемых в сетях Frame Relay, является то, что помимо указания параметров пропускной способности, указывается класс трафика. В сети ATM выделяется 5 классов трафика (см. табл. 7.4).

Таблица 7.4.

Классы трафика ATM

Класс трафика	Характеристика
A	Постоянная битовая скорость — Constant Bit Rate, CBR. Важны временные соотношения между передаваемыми и принимаемыми данными. С установлением соединения. Примеры: голосовой трафик, трафик телевизионного изображения.
B	Переменная битовая скорость — Variable Bit Rate, VBR. Важны временные соотношения между передаваемыми и принимаемыми данными. С установлением соединения. Примеры: компрессированный голос, компрессированное видеоизображение.
C	Переменная битовая скорость — Variable Bit Rate, VBR. Не важны временные соотношения между передаваемыми и принимаемыми данными. С установлением соединения. Примеры: трафик компьютерных сетей, в которых конечные узлы работают по протоколам с установлением соединений: frame relay, X.25, LLC2, TCP
D	Переменная битовая скорость — Variable Bit Rate, VBR. Не важны временные соотношения между передаваемыми и принимаемыми данными. Без установления соединения. Примеры: трафик компьютерных сетей, в которых конечные узлы работают по протоколам без установления соединений (IP, Ethernet DNS, SNMP).
X	Тип трафика не определен и полностью описывается количественными параметрами, задаваемыми пользователем (см. ниже).
нет	Если поддержание параметров пропускной способности и качества обслуживания для соединения неважно, то в запросе на установление соединения можно указать признак "Best Effort" ("по возможности"). Такой тип трафика получил название трафика с неопределенной битовой скоростью — Unspecified Bit Rate, UBR. Для трафика UBR сеть выделяет ресурсы "по возможности", то есть те, которые в данный момент свободны от использования виртуальными каналами, заказавшими определенные параметры качества обслуживания.

Помимо класса трафика, в трафик-контракте указываются и количественные параметры:
- Peak Cell Rate (PCR) — максимальная скорость передачи данных;
- Sustained Cell Rate (SCR) — средняя скорость передачи данных;
- Minimum Cell Rate (MCR) — минимальная скорость передачи данных;
- Maximum Burst Size (MBS) — максимальный размер пульсации;
- Cell Loss Ratio (CLR) — доля потерянных ячеек;
- Cell Transfer Delay (CTD) — задержка передачи ячеек;
- Cell Delay Variation (CDV) — вариация задержки ячеек.

Заключение трафик-контракта происходит автоматически, при установлении виртуального канала, по схеме аналогичной описанной для сети Frame Relay, используя пакет SETUP. Сходными с Frame Relay методами осуществляется и управление пропускной способностью сети: ячейки-нарушители трафик-контракта отмечаются признаком CLP=1 (Cell Loss Priority – приоритет потери кадра) и удаляются при перегрузке коммутаторов.

Передача трафика IP через сети ATM – протокол Classical IP

На основании технологии ATM можно построить полностью самодостаточные сети и передавать в ячейках ATM сразу пакеты протоколов прикладного уровня. Однако в реальности сети ATM чаще всего используются не самостоятельно, а как универсальный транспорт, позволяющий передавать трафик других сетей. При этом по сети ATM передаются пакеты протоколов канального и сетевого уровня других технологий: Ethernet, IP, IPX, Frame Relay, X.25, т.е. сеть ATM не заменяет старые технологии, а сосуществует с ними.

Рассмотрим как решается проблема передачи трафика IP-сетей через сети ATM. Для этих целей был разработан протокол Classical IP (RFC 1577). В соответствии со спецификацией Classical IP одна сеть ATM может быть представлена в виде нескольких логических IP-подсетей LIS (Logical IP Subnet), см. рис. 7.7. Все компьютеры одной LIS имеют общий IP-адрес сети. Как и в обычной IP-сети, прямые соединения на каналь-

ном уровне между компьютерами из разных LIS невозможны: такой трафик должен обязательно проходить через маршрутизатор, который и занимается доставкой пакета на сетевом уровне. Здесь необходимо отметить, что, теоретически, прямые соединения между компьютерами из разных LIS возможны, т.к. все они подключены к одной сети ATM, однако протокол Classical IP запрещает это делать, требуя, чтобы трафик между двумя разными LIS проходил только через маршрутизатор. Это позволяет логически структурировать сеть на более мелкие подсети и легче контролировать трафик между подсетями в привычной для системных администраторов форме – используя межсетевые экраны (firewall) на маршрутизаторе.

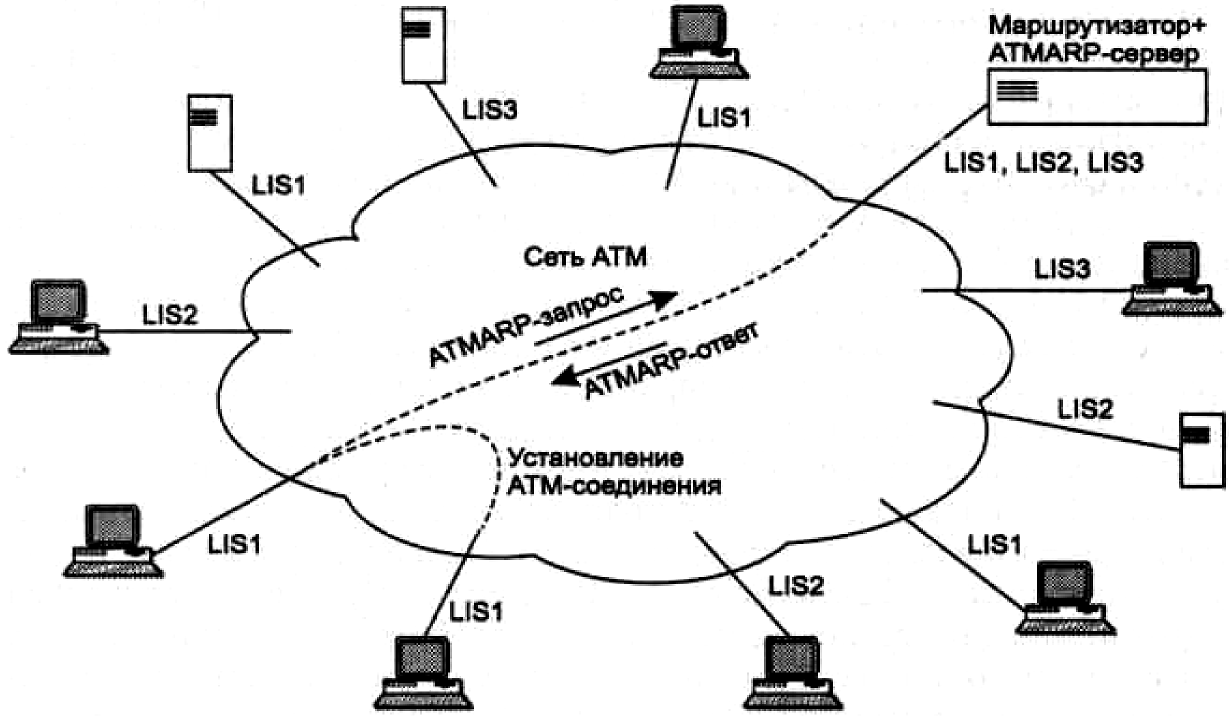


Рис. 7.7. Разделение сети ATM на логические IP-подсети (LIS) в протоколе Classical IP.

Маршрутизатором, в данном случае, является сетевое устройство, подключенное к сети ATM при помощи одного физического интерфейса, но этот физический интерфейс имеет несколько IP-адресов – по одному IP-адресу в каждой из LIS. Маршрутизатор также может быть совмещен с сервером ATMARP, который выполняет функции протокола ARP обычных IP-сетей (см. лекции ранее). В обычных IP-сетях протокол ARP отвечает за нахождение соответствия "IP-адрес компьютера" – "MAC-адрес сетевой карты компьютера" и работает широковещательно, т.е. ARP-запросы направляются "всем подряд" в расчете на то, что нужный компьютер распознает свой IP-адрес и сообщит свой MAC-адрес. В сети ATM широковещательные запросы не предусмотрены, поэтому для централизованного хранения информации о соответствии "IP-адрес компьютера" – "ATM-адрес компьютера" выделяется отдельный ATMARP-сервер, который строит свои таблицы автоматически. Если какой-либо компьютер обращается с запросом к ATMARP-серверу, то ему направляется встречный инверсный запрос ATMARP, чтобы выяснить IP- и ATM-адреса этого компьютера и зарегистрировать его в таблицах ATMARP-сервера.

Компьютеры конфигурируются традиционным образом: для них задается их собственный IP-адрес, маска подсети, IP-адрес маршрутизатора по умолчанию и ATM-адрес (или номер VPI/VCI для постоянного виртуального канала) сервера ATMARP. Если компьютер-отправитель хочет отправить пакет компьютеру-получателю из той же LIS, то он пошлет IP-адрес компьютера-получателя на сервер ATMARP, сервер просмотрит свою базу данных и вернет ATM-адрес компьютера-получателя, после чего компьютер-отправитель установит с компьютером-получателем прямое соединение, используя средства сети ATM. Если же компьютер-получатель находится в другой LIS (что видно по маске подсети), то пакет будет направлен не напрямую, а на маршрутизатор, который и займется дальнейшей доставкой пакета.

Использование технологии ATM в локальных сетях – спецификация LAN Emulation (LANE).
Рассмотренная выше схема Classical IP требует полной замены сетевого оборудования на оборудование ATM-сети. Это приемлемо в глобальных сетях, где основную стоимость составляют оптоволоконные линии большой длины, так что замена старого оборудования на коммутаторы ATM будет экономически оправдана. Однако в локальных сетях внедрение технологии ATM по затратам равнозначно созданию новой сети. Поэтому хотелось бы иметь возможность не полностью заменять уже купленное и работающее оборудование, а постепенно "внедрять" высокоскоростные коммутаторы ATM в уже работающую сеть. Такая возможность реализована в спецификации LANE (LAN Emulation - эмуляция локальных сетей).

Технология LANE позволяет на канальном уровне объединить между собой различные физические сегменты, при помощи коммутаторов ATM (см. рис. 7.8.). Необходимо отметить, что для протокола IP (или другого протокола сетевого уровня) такая "объединенная" сеть будет выглядеть как единый сегмент сети канального уровня. Для объединения нескольких сегментов LANE между собой на сетевом уровне необходимо использовать обычные маршрутизаторы локальных сетей.

В спецификации LANE предполагается, что каждый из физических сегментов сети подключен к коммутаторам ATM при помощи специальных конвертеров, которые преобразуют кадры и адреса Ethernet (или других протоколов канального уровня) в кадры и адреса ATM. В конверторы встроено специальное программное обеспечение LEC (LAN Emulation Client, клиент LANE). Также имеется сервер LES (LAN Emulation Server), который ведет общую таблицу, где указывается соответствие "MAC-адрес компьютера" – "ATM-адрес конвертора, через который к сети ATM подключен данный компьютер". Если компьютер-отправитель хочет направить пакет компьютеру-получателю из другого физического сегмента, то этот пакет попадет на конвертор (клиент LEC), который передаст на сервер LES MAC-адрес компьютера-получателя и запросит ATM-адрес конвертера, к которому подключен компьютер-получатель. После получения ATM-адреса конвертера - получателя, конвертер-отправитель установит с ним виртуальный канал средствами сети ATM и дальнейшее взаимодействие между компьютером-отправителем и компьютером-получателем будет идти через виртуальный канал и соответствующие конверторы, которые будут преобразовывать кадры

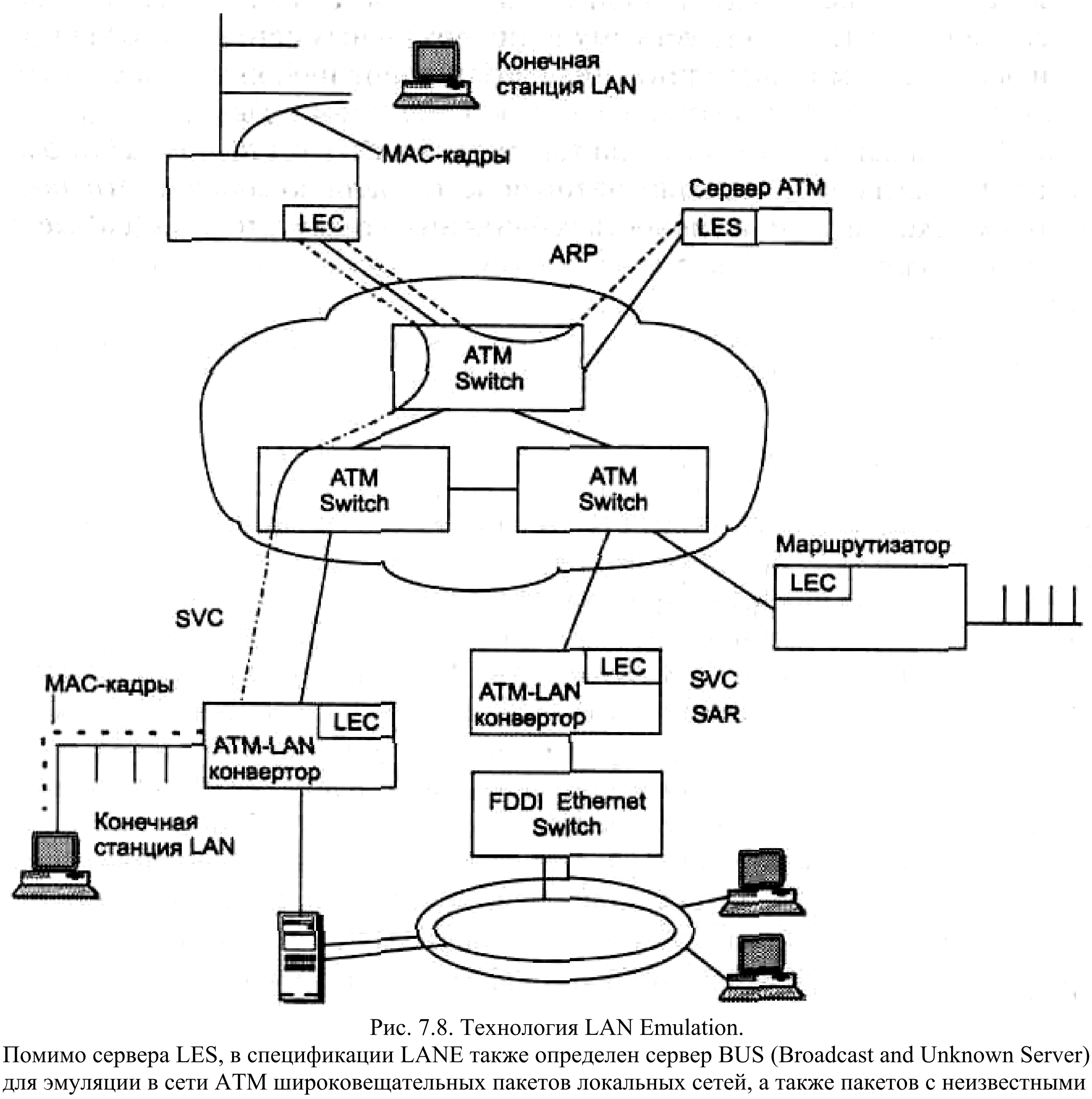


Рис. 7.8. Технология LAN Emulation.

Помимо сервера LES, в спецификации LANE также определен сервер BUS (Broadcast and Unknown Server) для эмуляции в сети ATM широковещательных пакетов локальных сетей, а также пакетов с неизвестными адресами. Этот сервер распространяет такие пакеты во все пограничные коммутаторы (и, соответственно, во все конверторы). Если на канальном уровне необходимо объединить между собой несколько эмулируемых сетей, аналогичных приведенным на рис. 7.8., то для каждой такой сети создаются собственные серверы LES и BUS, а в пограничных коммутаторах активизируют по одному элементу LEC для каждой эмулируе-

мой сети, а также вводят дополнительный сервер конфигурации LEGS (LAN Emulation Configuration Server) для хранения информации о количестве объединяемых эмулируемых сетей и об ATM-адресах серверов LES и BUS в каждой из этих сетей. Еще раз напомню, что для объединения нескольких эмулируемых сетей на сетевом уровне применяются обычные маршрутизаторы.

Будущее технологии ATM

Технология ATM внедряется не очень быстро, но планомерно: ежегодный прирост числа сетей ATM составляет 20-30 %. Сегодня основной потребитель коммутаторов ATM — это Internet. Сети ATM оказались более выгодной средой соединения IP-маршрутизаторов, чем выделенные каналы SDH, так как использование техники виртуальных каналов позволяет оплачивать только ту пропускную способность и качество обслуживания, которые действительно нужны (это же реализуется и в сетях Frame Rlay, но скорость передачи данных в них ниже, да и качество обслуживания проработано слабо).

В локальных сетях технология ATM применяется обычно на магистралях, где необходима высокая скорость передачи данных, однако тут у нее есть серьезный соперник — технология Gigabit Ethernet. Она превосходит ATM в скорости передачи данных — 1000 Мбит/с по сравнению с 622 Мбит/с, а также в денежных затратах на единицу скорости. Там, где требуется только высокая скорость передачи данных, технологию ATM, очевидно, заменит технология Gigabit Ethernet. Там же, где качество обслуживания важно (видеоконференции, трансляция телевизионных передач и т. п.), технология ATM останется. Для объединения настольных компьютеров технология ATM, вероятно, еще долго не будет использоваться, в основном из-за дороговизны.

7.2.4. Сети TCP / IP (сеть Internet).

Принципы построения и функционирования сетей TCP/IP уже подробно рассматривались ранее (см. лекции "Принципы функционирования ЛВС: протоколы и адресация" и "Сетевое оборудование"). Сразу хочется отметить, что, в отличие от всех рассмотренных выше сетей, сеть TCP/IP ***не работает*** с установлением виртуального канала. Маршрутизация осуществляется не на основании номера виртуального канала, а на основании полного IP-адреса, помещаемого в пакет. Протокол IP работает без установления предварительного соединения и не гарантирует доставку пакета. Подтверждение приема пакета и повторная передача утерянных пакетов – дело протоколов более высокого транспортного уровня (TCP), которые, тем не менее, все равно "передают" повторные пакеты "ненадежному" протоколу IP . Если в сетях ATM пропускная способность и качество обслуживания гарантированы, то в сети TCP/IP (IPv4) таких механизмов нет. Все пакеты сети TCP/IP доставляются в режиме, аналогичном режиму UBR (не задана битовая скорость, доставка "по возможности") сетей ATM.

Поскольку принципы функционирования сетей TCP/IP были достаточно подробно рассмотрены ранее, то здесь мы дадим только краткий обзор самой популярной глобальной сети TCP/IP – сети Internet. Причем обзор этот будет выполнен с точки зрения рядового пользователя, подключающегося к провайдеру Internet с домашнего компьютера, на котором установлен модем (dial-up access). Однако для начала немного истории.

Глобальная компьютерная сеть Internet начиналась с сети ARPAnet - оборонного проекта, который финансировался Агентством Перспективных Исследований Министерства Обороны США (Advanced Research Projects Agency, ARPA). Целю проекта являлась разработка компьютерной сети, призванной обеспечить устойчивое функционирование системы управления страной в условиях ядерной войны. В модели ARPAnet предполагалось, что любая часть сети может исчезнуть в любой момент. Несмотря на это сеть должна продолжать работать (насколько это возможно). Первые документы, описывающие технические требования к системе появились 1964 году, в 1969 первые четыре компьютера были объединены в реально действующую сеть ARPAnet. В 1971 году сеть насчитывала уже 14 компьютеров, а в 1972 г. - 37. В 1982 году были опубликованы протоколы Transfer Control Protocol (TCP) и Internet Protocol (IP). С этого момента и появился термин "TCP/IP".

Непосредственно сама сеть Internet появилась как результат большой компьютерной программы Национального Научного Фонда США (National Science Foundation, NSF). Для проведения научных исследований NSF организовал по всей стране несколько центров вычислений и оснастил их суперкомпьютерами, подключив к центрам вычислений американские университеты, и объединив все компьютеры в единую глобальную сеть. Первоначально планировалось использовать для этих целей ARPAnet, но администрация министерства обороны не разрешила подключение американских университетов к оборонной сети. В результате NSF создал свою собственную сеть NSFnet. В качестве основы этой сети были выбраны протоколы TCP/IP, разработанные в рамках проекта ARPAnet. Впоследствии к NSFnet присоединились еще порядка нескольких сотен различных сетей. Общим для всех этих сетей являлся тот факт, что для обмена информацией между собой они использовали единый механизм - семейство протоколов TCP/IP. Таким образом и зародился Internet – глобальная сеть, объединяющая локальные сети на основании протокола TCP/IP. В это время появились первые шесть доменов Internet: gov, mil, edu, com, org и net (gov - домен правительственных организаций, mil - домен военных организаций, edu - домен университетов, com - домен коммерческих

организаций, org - неправительственные и некоммерческие организации, net - домен организаций, отвечающих за функционирование самой сети).

Первоначально Internet существовал, как некоммерческая сеть, которая использовалась для обмена результатами научных исследований. Практически все лаборатории мира, имеющие доступ к Сети, стали размещать свои публикации в электронном виде в архивах Internet, а уже только после этого выпускать печатные копии этих работ. До 1989 года Internet оставалась некоммерческой сетью, и к ней подключались исключительно государственные и академические сети. В 1989 году к Internet подключилась первая коммерческая сеть – MCImail. В 1989 году в мире Internet произошла еще одна революция: Тим Бернерс-Ли (Tim Berners-Lee) создал язык гипертекстовой разметки (HTML), что привело к созданию в Internet нового сервиса – сети World Wide Web ("всемирная паутина", WWW). Фактически, большинство начинающих пользователей, перемещаясь по Internet (serfing), редко пользуются чем-то большим, чем гипертекстовые HTML-страницы, поэтому для них сеть WWW и есть Internet.

Самый простой способ получить навыки работы с Internet – это обратиться в Internet-кафе или на почту. Если дома имеется компьютер с внешним или внутренним модемом, то можно воспользоваться беспарольным доступом, предоставляемым АТС (в Минске – по тел. 8-600100). Общим у описанных выше способов является то – что они достаточно дороги. Гораздо выгоднее (и по-прежнему просто) приобрести Internet-карточку в киоске СоюзПечати, где будет указано количество часов доступа, которые дает карточка, а также имя и пароль пользователя (под защитным слоем) для подключения к сети. Однако самым выгодным, хотя и более трудоемким вариантом будет выбрать себе фирму-провайдера Internet, съездить в ее офис и заключить договор. Существует большое количество тарифных планов, предоставляющих различные варианты оплаты доступа в Internet. Среди студентов наиболее популярен "ночной" вариант неограниченного доступа (unlimited), когда взимается только абонентская плата, а число часов подключения не учитывается. Правда доступ осуществляется только в ночные часы (в некоторых тарифных планах и в выходные дни), а если фирма проводит политику низких цен и не расширяет свои модемные пулы, то дозвониться "в Internet" бывает невозможно. Также важно насколько "широкий канал" (в смысле пропускной способности) через который провайдер подключен к Internet. В противном случае, даже соединившись с провайдером на скорости 33,600 Кбит/с (максимум для аналоговых АТС), Вы не получите приемлемой скорости работы, т.к. параллельно с Вами на том же канале будут "сидеть" и другие пользователи. Поэтому выбор провайдера – самая трудоемкая часть в описанной схеме и чаще всего при этом руководствуются принципами "по совету друзей" и "по предыдущему опыту". После выбора схемы подключения к интернет (беспарольный доступ, Internet-карточка, договор с провайдером) все что необходимо сделать в ОС Windows – это создать новое соединение (см. "Пуск/Программы/…/ Удаленный доступ к сети"), указав телефон, имя пользователя и пароль. В большинстве случаев этого достаточно, однако возможно потребуется настроить параметры удаленного доступа: указать адреса DNS серверов и т.п. Для этого, на созданном соединении вызывается контекстное меню и в пункте свойства, в диалоговом окне, настраиваются необходимые параметры (подробные инструкции обычно указываются провайдером).

Web-браузеры

Как уже говорилось выше, большую часть пользователей в Internet интересуют гипертекстовые HTML-страницы, которые позволяют представить информацию в виде документов с перекрестными гиперссылками и привлекательным графическим оформлением. Для просмотра гипертекстовых страниц применяются специальные программы – Web-браузеры. На сегодняшний день существует большое количество Web-браузеров, но самыми популярными являются Internet Explorer (входит в состав ОС Windows) и Netscape Navigator. Распространение также получил браузер Opera. Остальные браузеры занимают незначительную долю рынка, менее 1%. Несмотря на то, что все браузеры предназначены для одного и того же – просмотра HTML-страниц – между ними имеются различия: страницы в Internet Explorer и Netscape Navigator выглядят по-разному, хотя общая структура страницы сохраняется. Особенно большие проблемы возникают при использовании в HTML-страницах программ, написанных на языке JavaScript, т.к. объектные модели Internet Explorer и Netscape Navigator различаются. Фактически приходится создавать два варианта страниц: один – для Internet Explorer, второй – для Netscape Navigator. Различаются браузеры и с точки зрения безопасности от сетевых атак. Хотя трудно возлагать всю вину только на браузер, но можно с уверенностью утверждать, что пользователь ОС Linux, использующий Netscape Navigator, будет иметь гораздо меньше проблем с безопасностью, чем пользователь ОС Windows 95/98/Me, использующий Internet Explorer :) . Для повышения защищенности, пользователям ОС Windows можно порекомендовать почаще скачивать с сервера Microsoft заплатки (patch, hotfix) и обновления (service pack), устраняющие выявленные пробелы в безопасности, а также завести себе межсетевой экран (например, AtGuard).

Поисковые системы

Для того, чтобы просмотреть HTML-страницу достаточно просто ввести ее URL-адрес в строке адреса Web-браузера, а затем следовать по гиперссылкам. Но именно в этом и заключается основная проблема – как узнать адрес страницы? Чаще всего бывает так, что известно то, что необходимо найти, но неизвестно где именно искать. Для решения этой проблемы существуют специальные поисковые системы. С точки зрения

пользователя, поисковая система – это обычный сайт на главной странице которого находятся разбитые по рубрикам ("Спорт", "Бизнес", "Компьютеры" и т.п.) ссылки на другие сайты. Кроме того, поисковая система позволяет пользователю ввести несколько ключевых слов и возвращает ссылки на страницы, содержащие эти ключевые слова. Важно отметить, что поиск не происходит в момент запроса пользователя. Отдельные серверы заранее и постоянно "исследуют" Internet и составляют базу данных по результатам поиска, а при поступлении запроса пользователя информация просто извлекается из этой базы данных. Из этой схемы имеется одно следствие: разные поисковые системы могут "исследовать" разные "области" Internet, поэтому если информация не найдена одной поисковой системой, то ее возможно найдет другая поисковая система. Кроме того, разные поисковые системы проводят поиск с разной эффективностью и на разную глубину. Самыми известными поисковыми системами по русским ресурсам Internet являются www.aport.ru, www.yandex.ru, www.rambler.ru. Наиболее известные поисковые системы по англоязычным ресурсам - www.altavista.com, www.yahoo.com, infoseek.go.com. Стоит также выделить поисковую систему www.google.com, которая достаточно быстро и качественно осуществляет поиск как по русским, так и по англоязычным ресурсам.

Как уже указывалось выше, все поисковые системы предусматривают поиск по ключевым словам. Очень важно правильно составить запрос на поиск. Необходимо употреблять ключевые слова комбинация которых не является широко распространенной. Если в ответ на Ваш запрос было найдено 7 321 сайт, то очевидно стоит попробовать другую комбинацию ключевых слов, т.к. у Вас просто времени не хватит просмотреть все сайты, большинство из которых не относится к делу. Практически в каждой поисковой системе имеется "расширенный поиск" (advanced search), который позволяет при помощи удобных форм и логических условий "и", "или" и шаблонов поиска организовать достаточно сложный поиск. Кроме того, каждая поисковая система имеет свой собственный язык запросов. К сожалению, единого стандарта не существует, поэтому просто приведем примеры поисковых запросов поисковой системы www.aprot.ru:

Таблица 7.5

Язык запросов поисковой системы www.aport.ru

Запрос	Результат поиска
пара умников	страницы содержащие слово "пара" и слово "умников". Для простых русских слов Aport также будет искать различные формы слов: умник, умников, умники, пара, пару, парой и т.д.
(пара) or (умников)	страницы содержащие слово "пара" или слово "умников".
"(пара) or (умников)"	страницы, содержащие слово "(пара)" и слово "or" и слово "(умников)". Условие or и другие специальные слова в кавычках игнорируются и считаются простым текстом.
((NOT из*) and (!яблоко)) or (шампунь))	Скобки означают порядок применения операторов "or" и "and". Звездочка означает любое количество произвольных символов, т.е. из* соответствует словам "изморозь", "известковый" и т.д. NOT – документ не должен содержать слово, следующее за этим оператором, т.е. (NOT из*) означает, что в странице не должны содержаться слова, начинающиеся на "из". Восклицательный знак означает "только эта форма слова", т.е. не будут учитываться слова "яблоки", "яблокам" и т.д.
сл7(курить грабли)	в найденных страницах, между словами "курить" и "грабли" должно быть не более 7 слов.
пр2(курить грабли)	в найденных страницах, слова "курить" и "грабли" должны находится в пределах 2-х предложений.
(пингвин) and (url=www.microsoft.com)	будут найдены все страницы на сервере www.microsoft.com, содержащие слово "пингвин"
(пингвин) and (url= *.ru/arktika/*)	будут найдены все страницы со словом "пингвин", при условии, что на некотором сервере из домена .ru, в основном каталоге web-сервера имеется подкаталог arktika.
(пингвин) and (date=01/01/98-01/02/99)	страницы, содержащие слово "пингвин" и созданные между 01.01.98 и 01.02.99
(пингвин) and (date:<01/02/99)	страницы, содержащие слово "пингвин" и созданные до 01.02.99

* Еще раз напомним, что данный язык запросов специфичен только для поисковой системы www.aport.ru. Другие поисковые системы имеют другие языки запросов – единый стандарт отсутствует.

Помимо поисковых систем можно воспользоваться некоторыми специализированными каталогами. Так для поиска программного обеспечения можно обратиться к сайтам www.listsoft.ru, www.tucows.ru, www.shareware.com. Для поиска художественной литературы можно порекомендовать библиотеку Максима Мошкова www.lib.ru.

Электронная почта

Электронная почта и ее протоколы уже рассматривались ранее (см. прикладные протоколы SMTP, POP3, IMAP). Так электронная почта позволяет не только обмениваться письмами, но и приложить (attach) к письму любой файл: графический файл, программа и т.д. При этом к одному письму может быть приложено несколько файлов (attachment), благодаря использованию стандарта MIME (Multipurpose Internet Mail Extension), который позволяет приложить к письму произвольное количество attachment-ов, разделяя разные файлы между собой при помощи специальной строки-разделителя (произвольный набор символов, который не встречается в файлах данных, и служит для указания границ файлов).

Адрес электронного почтового ящика вида vasya@server.ru можно получить двумя путями: первый – завести себе платный почтовый ящик на каком-либо сервере (в частности, у своего провайдера), второй – получить бесплатный почтовый ящик на одном из серверов в Internet. Существует большое количество серверов, которые позволяют создать (sign in) собственный бесплатный почтовый ящик ограниченного объема, просто заполнив несколько простых форм (не обязательно указывать реальные данные). Приведем примеры адресов: www.hotmail.com, www.yahoo.com, www.mail.ru, www.tut.by, www.torba.com и др. Работать с такими почтовыми ящиками можно по протоколу http при помощи обычного Web-браузера (например, Internet Explorer) или, если сервер предоставляет конкретный вид сервиса, по протоколам SMTP или POP3, при помощи специальных программ почтовых-клиентов Outlook Express, Microsoft Outlook, Netscape Communicator, The Bat.

Программа-пейджер ICQ

В Internet существует большое количество интерактивных чатов – сайтов, где можно в реальном времени, при помощи клавиатуры, пообщаться с другими людьми, также зашедшими на этот сайт. Одним из таких популярных сайтов является www.icq.com. Израильская фирма Mirabilis, поддерживающая этот сайт, создала специальную программу ICQ для расширения возможностей интерактивных чатов. Название программы ("ICQ") происходит от игры слов "I Seek You" - "я ищу тебя". В русском варианте программа получила неофициальное имя "Аська". ICQ фактически является виртуальным пейджером. Достаточно скачать с сайта www.icq.com или найти на CD-диске программу ICQ. В процессе установки программы пользователь регистрируется в базе данных Mirabilis и получает индивидуальный номер пользователя (UIN, User Identification Number), который имеет такой же смысл, что и номер обычного пейджера. При помощи своего экземпляра программы ICQ, любой человек (при условии, что вы это ему разрешите) может направить Вам сообщение. Если Вы активны в это время (подключены к Internet и запустили программу ICQ), то получите это сообщение немедленно и сможете направить ответ. Если же Вы отключены от Internet, то это сообщение останется в базе данных Mirabilis, и, когда Вы следующий раз подключитесь к Internet и запустите программу ICQ, это сообщение будет доставлено Вам. Программа ICQ имеет много дополнительных возможностей. Например, если имя Вашего знакомого в окне программы ICQ отмечено синим цветом (цвет зависит от настроек), то это значит, что Ваш знакомый сейчас находится в Internet. Если же имя отмечено красным цветом, то это значит, что Ваш знакомый либо не подключился к Internet, либо не запустил программу ICQ. При помощи ICQ можно также отправлять короткие текстовые сообщения на сотовые телефоны, пересылать файлы и многое другое.

Создание и размещение собственных Web-страниц в Internet.

Собственный Web-сайт можно разместить на каком-либо сервере платно (например, у провайдера), или на одном из серверов Internet, предоставляющих возможность бесплатного размещения сайта. Примером такого бесплатного сервера является сервер narod.yandex.ru. При регистрации на сервере необходимо указать название создаваемого сайта, пароль и имя пользователя, краткие собственные данные (необязательно правдивые). После этого Вы получаете возможность создать свой сайт с именем вида "nazvanie.narod.ru". На сайте удобная система форм, которая позволяет создать собственный сайт по шаблону, самому набрать html-код страницы или загрузить страницы на сайт со своего компьютера по http или ftp. Другим известным сервером, предоставляющим бесплатный хостинг (размещение) Web-сайтов, является поисковый сервер www.yahoo.com (адрес geocities.yahoo.com/home/). Существует также большое количество других серверов, предоставляющих бесплатный Web-хостинг, каждый из которых отличается условиями размещения Web-страниц, "скоростью" самого сервера и другими параметрами. Рядом преимуществ обладает и платное размещение Web-страниц (см. табл.7.6)

Таблица 7.6.

Преимущества и недостатки бесплатного и платного размещения Web-страниц

Платное размещение Web-страниц	Бесплатное размещение Web-страниц
1. Условия размещения сайта регулируются договорными отношениями. Стороны несут ответственность за нарушение договорных условий.	1. Условия размещения сайта достаточно жесткие и зависят только от доброй воли администраторов сервера и могут произвольно изменяться в любой момент, вплоть до полного отказа в хостинге или существенного сокращения объема дискового пространства, предоставляемого для размещения сайта.
2. Объем предоставляемых сервисов (CGI, использование базы данных через SQL и т.д.) определяется договорными условиями.	2. Список предоставляемых сервисов сильно ограничен. Сервисы CGI и баз данных обычно не предоставляются.
3. Объем дискового пространства, предоставляемого для размещения сайта ограничен только договорными условиями.	3. Объем дискового пространства, предоставляемого для размещения сайта обычно невелик и одинаков для всех.

При использовании бесплатного Web-хостинга часто возникает следующая проблема: допустим Вы разместили сайт на бесплатном сервере и получили URL-адрес вида "www.halyava.com\pub\free\html\56371sait\". Вряд ли руководство фирмы устроит такой адрес, который во-первых трудно запомнить, а во-вторых свидетельствует о "несерьезности" сайта, размещенного на явно бесплатном сервере. Или, например, вполне реальна ситуация, когда на сервере narod.ru нужное Вам название сайта уже занято кем-то другим.

Для решения этой проблемы можно приобрести доменное имя (см. лекцию по DNS). Например, можно заплатить за регистрацию в домене ".by" и получить URL-адрес, который фактически будет указывать на ту же самую страницу на бесплатном сервере, но будет иметь вид "www . что_угодно . by". Если нет желания платить за доменное имя, то можно воспользоваться бесплатной регистрацией доменного имени вида "ваш_сайт . da . ru " на сервере www.da.ru. При "переезде" Вашего сайта на новый сервер, достаточно изменить тот адрес, куда указывает зарегестрированное доменное имя, и для Ваших пользователей "переезд" пройдет незамеченным. Как альтернативный вариант можно разместить на старом адресе ссылку, которая сообщает о переезде сайта и, средствами JavaScript, перенаправляет пользователя на новый сайт.

Лекция 8. Язык HTML, DHTML и CSS.

Язык гипертекстовой разметки HTML (HyperText Markup Language) был предложен Тимом Бернерсом-Ли в 1989 году и в настоящее время является стандартом для представления гипертекстовых документов в сети World Wide Web (WWW). Поскольку большинство людей, перемещаясь по сайтам при помощи гиперссылок, никогда не покидают WWW, то можно сказать, что HTML – это один из основных языков в Internet.

Язык HTML представляет собой инструкции, называемые тэгами, на основании которых Web-браузер (например Internet Explorer) создает и форматирует гипертекстовую страницу. Тэг состоит из трех элементов:

Таблица 8.1.

Структура тэга

Элемент	Синтаксис	Пример
1. Начало тэга	<тэг параметр1=значение1 параметрN=значениеN>	<TABLE WIDTH=50>
2. Тело тэга	содержимое, зависит от самого тэга	
3. Конец тэга	</тэг>	</TABLE>

Поскольку тэги распознаются и выполняются web-браузером, то язык HTML не зависит от типа компьютера. Существует множество различных тэгов, позволяющих включать в HTML-страницу таблицы, рисунки, гиперссылки, задавать шрифт и цвет фона, и даже встраивать в страницу программы, написанные на языках Java, Java-script и VBScript. Однако для того, чтобы тэги работали, они должны быть правильно организованы. Браузеры не выполняют неизвестные или не правильно записанные тэги, благодаря чему достигается надежность HTML: даже если вы все сделаете не правильно – взрыва не произойдет, просто правильные команды будут выполнены, а не правильные или неизвестные браузер проигнорирует.

Ниже будут рассмотрены некоторые основные тэги HTML-документа. Конечно, можно не зная ни одного тега создать HTML-документ при помощи специализированных программ, таких как ***MS Front Page***, или, на худой конец, создать документ Word с гиперссылками и сохранить его в формате HTML. Однако полезно знать хотя бы минимум информации по языку HTML, что повысит эффективность работы (т.к. специализированные программы порой создают крайне не эффективный HTML-код) и позволит использовать чужие HTML-странички для разработки собственных. Ведь для того, чтобы просмотреть HTML-код понравившегося сайта, достаточно дать команду "Просмотр в виде HTML" в контекстном меню Internet Explorer и подправить его при помощи "Блокнота" или любого другого текстового редактора, позволяющего сохранять файл в формате "Только текст". Базовая структура HTML-документа представлена ниже:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>  Это заголовок HTML-документа, однако!            </TITLE>
</HEAD>
<BODY>  А это содержимое HTML-документа, однозначно !!!   </BODY>
</HTML>
```

Здесь тэги обозначают следующее:

Таблица 8.2.

Тэги структуры HTML-документа.

Тэг	Значение
<!DOCTYPE HTML PUBLIC "-//W3C// DTD HTML 4.0 Transitional//EN">	Тег указывает на то, что данный документ является документом HTML и указывает версию HTML. Фактически, этот тег является комментарием HTML и не обязателен. Однако, учитывая существование документов XML, его желательно указывать.
<HTML> </HTML>	Указывают на начало и конец HTML-документа. Любой HTML-документ должен быть заключен в эти тэги. В тэге HTML иногда указывается версия языка HTML использованная при написании этого документа.
<HEAD> </HEAD>	Начало и конец заголовка документа. В заголовке обычно указывается различная служебная информация: ключевые слова, тип кодировки документа и т.д. Это поле обычно просматривают поисковые сервера для того, чтобы определить содержимое документа.
<TITLE> </TITLE>	Текст, заключенный между этими тэгами, отображается в заголовке окна Internet Explorer при просмотре HTML-документа.

Тэг	Значение
<BODY> </BODY>	Содержимое HTML-документа. Тэг BODY может иметь следующие параметры: text – цвет текста документа. Название цвета указывается английским словом или в RGB. Пример: <BODY text="yellow"> или text="#FFFF00". background – фоновая картинка документа. Пример: <BODY background="fon.gif">. bgproperties – при задании этому параметру значения fixed, фоновая картинка не будет прокручиваться вместе с документом, т.е. будет неподвижна. Пример: <BODY bgproperties="fixed"> bgcolor – цвет фона документа. Если указать одновременно и параметр background и параметр bgcolor, то цвет фона документа будет отображаться только если не удастся загрузить фоновую картинку. nowrap– при задании этого параметра, строчка, не помещающаяся в окне, не будет переносится на новую строку (появятся полосы прокрутки). Пример: <BODY nowrap> link – цвет гиперссылок в документе. alink – цвет активных (выделенных) гиперссылок в документе, vlink – цвет посещенных гиперссылок в документе.

В заголовке, при помощи тэга <META> полезно указать ключевые слова (keywords) и кодировку (charset) документа. Ключевые слова используются поисковыми машинами Internet, при анализе содержимого документа. Кодировка важна для правильного отображения языка документа. В рамках тэга <BODY> </BODY> пишется содержимое всего остального документа, в виде обычного текста, заключенного в тэги форматирования, а также другие специальные тэги.

Пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Пример HTML-документа</TITLE>
<META name="keywords" content="ключевое слово1, ключевое слово2, ключевое слово3">
<META content="text/html; charset=windows-1251" http-equiv="Content-Type">
</HEAD>
<BODY text="blue" bgcolor="yellow" background="fon.gif" bgproperties="fixed"  link="green"  alink="red"
nowrap>
<!--  Это комментарий. Отображаться на экране не будет  -->
<B>      Этот текст будет выведен жирным (Bold) шрифтом     </B>
<CENTER><I><B><U>   Этот текст будет выведен по центру (Center), наклонным (Italic), жирным (Bold) и
подчеркнутым (Underline) шрифтом    </U></B></I></CENTER>
<H1> Это текст заголовка (Heading), самый крупный </H1>
<H6> Это текст заголовка (Heading), самый мелкий </H6>
<FONT  face="Times" size="7" color="white">  Этот текст будет выведен шрифтом Times, размером 7
(размеры от 1 до 7), белым цветом </FONT>
<BR> Здесь будет новая строка (BR = Break). Тэг BR закрывать не надо.
<HR size="5px" width="70%" align="left" > Здесь будет горизонтальная линия: толщина=5 пикселей, дли-
на=70% экрана (можно в пикселях), выравнивание – по левому краю страницы.  Тэг HR закрывать не надо.
<PRE> Текст, расположенный между тэгами PRE, будет выводится на экран "как есть", т.е. с учетом вводов,
пробелов и т.п. </PRE>
<A HREF="http://www.microsoft.com">   Этот текст будет гиперссылкой     </A>
<A NAME="Metka">   Этот текст будет помечен закладкой (Anchor) с именем Metka   </A>
<A HREF="#Metka">  Вот так, внутри документа, можно сделать гиперссылку на созданную выше закладку
Metka  для быстрого перехода по тексту документа  </A>
<A HREF="../../katalog/file.html#Metka3"> Вот так можно сделать гиперссылку на конкретное место
(закладку Metka3) файла file.htm, находящийся на два каталога выше, в папке katalog  </A>
<IMG src="http://microsoft.com/risunok.gif" alt="Тэг IMG отображает рисунок из файла risunok.gif, размером
300x200 пикселей, без рамки (рамка=0), а текст, который вы читаете, будет отображен, если рисунок не
удастся загрузить" width="300px" height="200px" border="0">
<!-- А вот так (см. ниже) можно сделать гиперссылку в виде рисунка.  -->
<A HREF="http://www.microsoft.com"> <IMG src="risunok.gif"> </A>
</BODY>
</HTML>
```

Создание таблиц в HTML

Среди других тэгов отдельного внимания заслуживает тэг TABLE, отвечающий за создание таблиц. Таблицы можно использовать как непосредственно, так и для привязки текста к определенному месту экрана. Так, например, обтекание рисунка текстом справа, в рамках HTML проще всего сделать, создав 2

ячейки таблицы: в левой – рисунок, в правой – текст, толщина рамки - 0. Ниже, для примера, приведен HTML-код и показана таблица, которую он создает:

Декабрь						
Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
1	2	3	4	5	6	7
8	9	10	11	12	13	14

рис 8.1. Пример создаваемой таблицы.

```
<TABLE width="500px" border="5" align="Left">
<TBODY>
<TR>  <TH align="Center" valign="Middle" colspan="7"> Декабрь  </TH>  </TR>
<TR Align="Center">
<TH>Понедельник</TH>  <TH>Вторник</TH>  <TH>Среда</TH>  <TH>Четверг</TH>
<TH>Пятница</TH>  <TH>Суббота</TH> <TH>Воскресенье</TH>  </TR>
<TR valign="Top">
<TD>1</TD> <TD>2</TD> <TD>3</TD> <TD>4</TD> <TD>5</TD> <TD>6</TD> <TD>7</TD> </TR>
<TR valign="Top">
<TD>8</TD> <TD>9</TD> <TD>10</TD> <TD>11</TD> <TD>12</TD> <TD>13</TD> <TD>14</TD>
</TR>
</TBODY>
</TABLE>
```

Здесь тэги и параметры тэгов имеют следующее значение:

Таблица 8.3.

Тэги создания таблицы.

Тэг	Значение
<Table> </Table>	начало и конец таблицы.
<TBODY></TBODY>	тело таблицы
Border	толщина рамки таблицы.
Width	ширина таблицы либо в пикселях, либо в процентах ко всему пространству.
Align	выравнивание содержимого ячеек (right, left, center).
<TR> </TR>	начало и конец строки таблицы.
<TH> </TH>	начало и конец ячейки шапки таблицы.
<TD> </TD>	начало и конец обычной ячейки таблицы.
Valign	вертикальное выравнивание содержимого ячеек (bottom, middle, top).
Colspan	количество столбцов, объединяемых данной ячейкой.
Rowspan	количество строк, объединяемых данной ячейкой.

Создание форм в HTML

Наибольший интерес для разработчика Web-страниц представляют формы. Заказ книги через Internet-магазин, регистрация на почтовом сервере, запрос на поиск информации – все это требует заполнения разнообразных форм. Internet просто переполнен различными формами. Ниже, в качестве примера, приведен HTML-код формы и показан ее внешний вид.

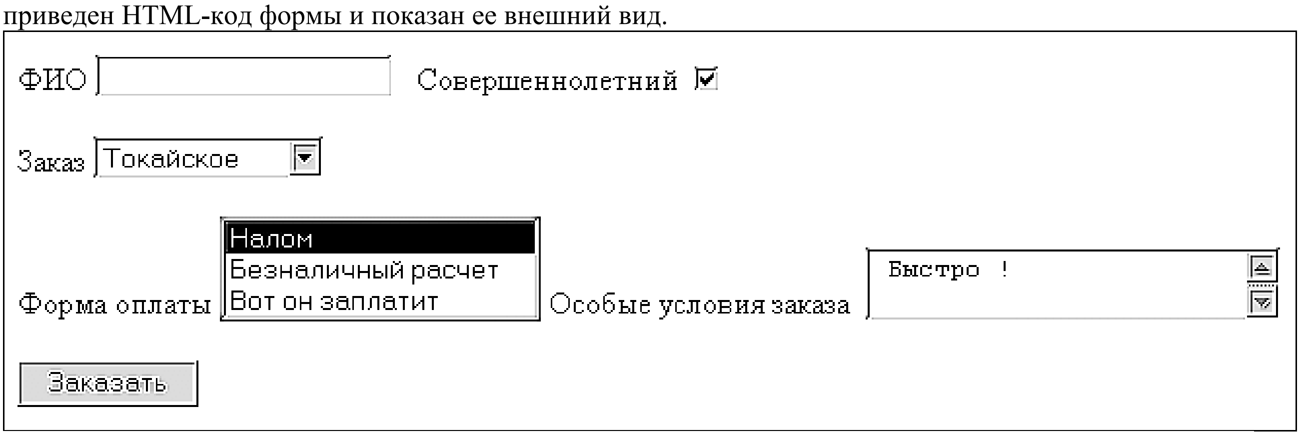


рис. 8.2. Пример создаваемой формы.

```
<FORM method="post" action="http://vino.com/zakaz.asp" name="primer">
<P>ФИО <INPUT type="text" name="FIO" size="20">
    Совершеннолетний <INPUT type="checkbox" name="SovershLetn" value="ON" checked></P>
<P>Заказ <SELECT size="1" name="zakaz">
   <OPTION value="Спирт1">Шато Лафит</OPTION>
   <OPTION value="Спирт2" selected>Токайское</OPTION>
   <OPTION value="Спирт3">Дон Пириньон</OPTION>
</SELECT> </P>
<P>Форма оплаты <SELECT size="3" name="Raschet" multiple>
   <OPTION selected>Налом</OPTION>
   <OPTION>Безналичный расчет</OPTION>
   <OPTION>Вот он заплатит</OPTION>
</SELECT> </P>
Особые условия заказа <TEXTAREA rows="2" name="Yslovia" cols="24"> Быстро !</TEXTAREA>
<P><INPUT type="submit" value="Заказать" name="OKbutton"></P>
</FORM>
```

Здесь тэги и параметры тэгов имеют следующее значение:

Таблица 8.4.

Тэги создания формы.

Тэг	Значение
<FORM> </FORM>	Начало и конец формы
Method	Определяет метод, используемый для передачи данных на сервер. Если применяется метод GET, то информация, внесенная в форму пользователем, передается на сервер как часть URL-адреса, в заголовке HTTP-запроса , например: http://vino.com/zakaz.asp?FIO=Иванов&SovershLetn=ON&ZAKAZ=Спирт1&Raschet=Налом Если используется метод POST, то информация занесенная в форму, передается на сервер в теле HTTP-запроса.
Action	Адрес программы на сервере, для которого предназначены занесенные в форму данные. Обычно это какая-то CGI, ASP или ISAPI программа, способная принимать отправляемую пользователем информацию и обрабатывать ее.
Name	Имя формы в HTML-документе. Необходимо для обращения к форме по ее имени из программы на VBScript или JavaScript.
<P> </P>	Начало и конец абзаца.
<INPUT> </INPUT>	Начало и конец элемента управления формы. Имеет следующие параметры: TYPE – указывает один из следующих типов элементов управления: checkbox (флажок), radio(кружок-переключатель), text(поле для ввода текста), password(введенные символы заменяются на ****), hidden(скрытое поле), button(обычная кнопка), reset(кнопка для очистки формы), submit(кнопка для отправки формы на сервер), image(кнопка submit в виде рисунка, при отправке формы передаются координаты XY места рисунка на котором щелкнул пользователь). NAME – имя элемента управления, SIZE – длина элемента в символах, VALUE – значение, выбранное пользователем.
<SELECT> </SELECT>	Задает список/выпадающий список. Имеет следующие параметры: NAME – имя списка, VALUE – значение, выбранное пользователем, SIZE - количество значений, одновременно отображаемых в списке, MULTIPLY – позволяет выбрать из список сразу несколько значений. <OPTION> </OPTION> - элемент списка. <OPTION SELECTED> </OPTION> - элемент списка, выбранный по умолчанию. <OPTION DISABLED> </OPTION> - отключенный элемент списка, отображается серым цветом.
<TEXTAREA> </TEXTAREA>	Область с полосами прокрутки для ввода большого количества текста. Имеет следующие параметры: NAME – имя области текста, ROWS - высота области (строк), COLUMNS - ширина области (колонок), WRAP - способ переноса текста в документе (off - отключен, virtual – перенос отображается только при вводе в форму, реально символы конца строки не вставляются, physical – включен).

Фреймы в HTML

Фреймы делят HTML-страницу на несколько независимых областей, каждая из которых функционирует независимо и может отображать свой URL-адрес. Возможен также случай, когда фрейм представлен в виде отдельного окна. Другими словами, фреймы – это попытка сделать HTML-страницу многооконной.

Для того, чтобы страница HTML-страница могла содержать фреймы, тэги <BODY> </BODY>, обрамляющие тело HTML-документа должны быть заменены на тэги <FRAMESET></FRAMESET>. Ниже приведен текст HTML-страницы, содержащей фреймы, и даны пояснения некоторых тэгов.

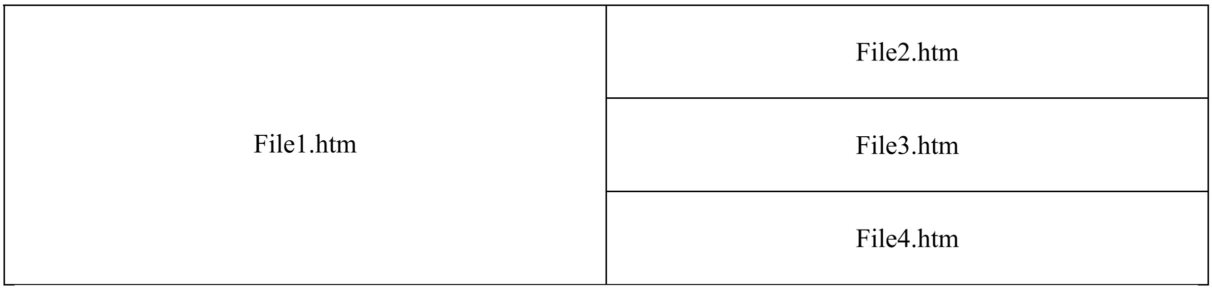


рис. 8.4. Страница фреймов.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html> <head> <title> Страница фрэймов </title> </head>
<frameset cols="45%,*">
      <frame name="levo" src="File1.htm" scrolling="no" frameborder="NO" noresize>
            <frameset rows="100px,200px,*">
               <frame name="verx"        src="File2.htm" scrolling="auto" frameborder="NO">
               <frame name="seredina"   src="File3.htm" scrolling="yes">
               <frame name="niz"          src="File4.htm" scrolling="yes">
            </frameset>
      <noframes>
      <body>  Эта страница использует фреймы, однако Ваш браузер их не поддерживает.  </body>
      </noframes>
</frameset>
</html>
```

Значение некоторых тэгов пояснено ниже:

Таблица 8.5.

Тэги создания фреймов.

Тэг	Значение
<frameset> </frameset>	Контейнер фреймов. Параметры ROWS и COLS задают разбиение страницы или по вертикали (COLS=45%,* указывает на 2 части: одна 45% ширины экрана, вторая – все остальное пространство *) или по горизонтали (ROWS=100,200,* указывает на 3 части: одна высотой 100 пикселей, вторая 200 пикселей и третяя – все остальное). С помощью контейнера экран можно разбить по вертикали или по горизонтали. Если надо одновременно разбить экран и по вертикали и по горизонтали, то лучше всего вложить один контейнер в другой, как это показано в примере.
<frame>	Задает URL-адрес файла, отображаемого в контейнере и его параметры. Name – имя фрейма, Src – URL-адрес файла фрейма, Scrolling – наличие полос прокрутки (yes, no, auto), Noresize – запрещает изменять размер фрейма, FrameBoder – задает отображение рамки фрейма (yes, no).
<noframes> </noframes>	Данные между этими тэгами предназначены для браузеров, которые не могут обрабатывать фреймы. В примере, в этих тэгах помещен текст, который увидят пользователи, если их браузер не воспринимает фреймы.

Использование разделенного рисунка

Разделенные рисунки — это один самых распространенных способов перемещения по страницам Web-узла. Они представляют собой рисунки, разделенные на несколько областей, каждая из которых выступает в роли отдельной ссылки. Щелчок на каждой из областей одного рисунка приводит к выполнению разных команд сценария. Существует два типа разделенных рисунков — клиентские и серверные. Клиентские разделенные рисунки определяют области ссылок на изображении в документе HTML и создаются с помощью HTML. Серверные разделенные рисунки требуют создания специального файла определения карты разделения рисунка, которым управляет Web-сервер. Здесь будут рассматриваться клиентские разделенные рисунки.

Чтобы создать клиентский разделенный рисунок, сделайте следующее.

1) В графическом редакторе создайте рисунок в формате GIF или JPEG.
2) Задайте карту разбиения рисунка на области в формате:

```
<MAP name="Имя_карты_разбиения">
<AREA SHAPE="rect" COORDS="x1, y1, x2, y2" HREF="http://adress1.com">
<AREA SHAPE="rect" COORDS="x1, y1, x2, y2" HREF="http://adress2.com">
<AREA SHAPE="rect" COORDS="x1, y1, x2, y2" HREF="http://adressN.com">
</MAP>
```

3) Включите рисунок в страницу, указав, что для него необходимо использовать созданную вами карту:

```
<IMG SRC="file.gif" alt="пример карты" USEMAP = "#Имя_карты_разбиения">
```

Пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><BODY>

<MAP NAME="karta1">
<AREA SHAPE=RECT COORDS="1,1,100,300" HREF="http://magazin.com/zal">
<AREA SHAPE=RECT COORDS="110,1,200,300" HREF="http://magazin.com/sklad">
</MAP>

<H1>Пример клиентского разделенного рисунка</H1> <BR>
<IMG SRC="file1.gif" alt="пример карты" USEMAP="#karta1" WIDTH="200px" HEIGHT="300px">
</BODY> </HTML>
```

Создание бегущей строки при помощи тэга <MARQUEE>.

Используя тэг <MARQUEE> можно просто создать бегущую строку, например так:

```
<marquee> Приветик </marquee>
```

Или в более полном варианте:

```
<marquee width="75%" height="20px" bgcolor="yellow" direction="left" behaviour="scroll" loop="10"
scrollamount="30px" scrolldelay="30" align="top"> Приветик </marquee>
```

Таблица 8.6.

Тэг marquee.

Параметр	Значение
width, height, bgcolor	Соответственно ширина, длина и цвет фона окна бегущей строки.
direction	Направление движения: left – слева, right – справа.
behaviour	Поведение: scroll – прокрутить текст n-раз и исчезнуть, slide-прокрутить текст n-раз и остаться, alternate – отскакивать от краев экрана.
loop	Число раз прокрутки. Если loop не указано, или указано число –1, то прокрутка будет бесконечной.
scrollamount	Количество пикселей, на которое строка смещается за 1 шаг.
scrolldelay	Задержка в милисекундах перед каждым шангом прокрутки.
align	Выравнивание строки в своем окне: top - верх, middle – центр, bottom - низ.

Слои DHTML, каскадные таблицы стилей CSS.

Традиционный HTML позволяет описать структуру документа, но не указывает "как" должен выглядеть документ. Тот или иной браузер сам решает каким шрифтом отобразить, например, заголовок первого уровня (тэг <H1>). Гораздо большего контроля над внешним видом документа можно добиться, используя каскадные таблицы стилей (CSS). Они позволяют явно задать цвет, размер, название и начертание шрифта, фон, тип рамки и т.п., для какого-либо фрагмента документа. Стили описываются при помощи тэга <STYLE>. Стили также могут быть описаны в отдельном файле (обычно, с расширением CSS), подключаемом к HTML-документам при помощи ссылки вида <link rel="stylesheet" type="text/css" href="файл_стилей.css">. В подключаемом файле тэги <STYLE> </STYLE> указывать не надо.

Помимо стилей, в DHTML таже предусмотрено существование слоев в документе. Слои чем-то напоминают листы кальки, наложенные друг на друга, или слои в Photoshop и др. графических редакторах. Слои могут быть наложены друг на друга в определенном порядке и перекрываться. С помощью JavaScript их также можно перемещать по экрану, скрывать и отображать. К каждому слою может быть применен уникальный стиль CSS. Слои описываются при помощи тэга <DIV>. Рассмотрим работу со стилями и слоями подробнее.

Стили описываются при помощи тэга <STYLE>, который может находиться в заголовке документа, теле документа, или входить как параметр в состав другого тэга. Стиль может описываться для какого-либо тэга, для слоя в документе, или описываться как "класс", который может быть применен к любому тэгу в документе или части текста, при помощи вспомогательного тэга <SPAN>. В нормальном состоянии стили каскадно "спускаются" по странице, т.е. если один тэг вложен в другой, то он наследует стили, определенные в "вышестоящем" по уровню вложенности тэге, если только эти стили не были переопределены в самом вложенном тэге. Тэг <SPAN> "отсекает" каскадное наследование стилей и создает в документе "вложенный контейнер", внешний вид содержимого которого полностью определяются стилями, примененными в тэге <SPAN>. Пример:

Таблица 8.7.

Пример использования стилей CSS и слоев DHTML.

Текст HTML-страницы	Комментарий
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <HTML> <HEAD> <TITLE> Слои DHTML и стили CSS </TITLE>	Начало HTML-документа и области заголовка, название HTML-документа.
<STYLE>	Начало описания стилей документа.
BODY { background-color : #EEEEEE; color : #000000; font-size:14px; font-family : Arial, Helvetica, sans-serif; }	Описания стиля для тэга BODY. Сначала указывается название тэга, для которого описывается стиль, а затем в фигурных скобках указывается название параметра, двоеточие, значение параметра, точка с запятой, следующие параметры. В анном тэге указан цвет фона #EEEEEE, цвет текста #000000, размер шрифта 14 пикселей, название шрифта Arial, если его не будет, то используется шрифт Helvetica и, в самом крайнем случае, стандартный шрифт браузера sans-serif. Данный стиль будет применяться к содержимому документа, дополняя (если это возможно и стили не переопределены в самих элементах), стили других элементов.
H1, FONT {font-size:110%; font-style:italic; font-weight:bold; }	Описание стиля сразу для двух тэгов: для заголовков первого уровня (H1) и тэга FONT. Размер шрифта 110%, наклонный (italic) и жирный (bold) шрифт.
A:HOVER, INPUT {color : #FF009F; text-decoration : underline; }	Стиль для гиперссылки (при наведению на нее мышью) и такой же стиль для элементов управления INPUT (кнопки, поля ввода, флажки и т.п.). Цвет #FF009F, текст подчеркнутый (underline).
A:LINK, A:VISITED {color : #000099; text-decoration : none; }	Стиль для гиперссылки (LINK) и для посещенной гиперссылки (VISITED). Цвет #000099, текст без подчеркивания (none).
#s1 {position:absolute; left:10px; top:300px; width:200px; height:100px; visibility:show; zIndex:1;}	Стиль для слоя (тэг <DIV>) с названием (id), равным s1. Слой описан ниже в тексте документа. Координаты абсолютные (absolute), слева – 10 пикселей, сверху 300, ширина 200, высота 100, видимость слоя – отображать (может быть hidden - скрытый), порядок наслоения – самый нижний слой (чем больше число – тем выше слой). Описание стиля слоя начинается со знака "#".
. zagolovok {color : #FF009F; font-size:large;}	Описание "класса" стиля. Класс может многократно применяться в документе, при помощи параметра class, добавляемого внутрь любого тэга. Описание класса начинается с точки.
</STYLE>	Конец области описания стиля.
</HEAD><BODY>	Конец области заголовка, начало тела документа.
<H1> Новый заголовок первого уровня </H1>	Этот заголовок первого уровня выглядит нестандартно, т.к. для него определен стиль.
<H1 class="zagolovok"> К заголовку первого уровня применен класс zagolovok </H1>	Этот заголовок первого уровня выглядит и не стандартно, и не так, как указано в стиле для тэга <H1>, т.к. к нему дополнительно применен стиль класса zagolovok (точка не указывается).
<A href="www.sait.com"> Гиперссылка </A>	Гиперссылка выглядит не стандартно, в соответствии с описанием в стиле.

Продолжение табл. 8.7.

Текст HTML-страницы	Комментарий
<form method="post" action="www.sait.by/cgi-bin/zakaz.cgi">	Начало формы.
<input type="text" size="20" name="FIO"> 	Поле ввода. Вводимый текст будет красным и подчеркнутым, т.к. стиль для гиперссылки (при наведении на нее мышью) совпадает со стилем элемента <INPUT>.
<input type="reset" value="Cancel" style="color:blue; font-weight:bold;">	Здесь стиль определен прямо в самом тэге. Причем это описание суммируется с описанием стиля для тэга <INPUT>, находящемся в заголовке документа. В результате, текст на кнопке будет синий (blue), жирный (bold) и подчеркнутый (унаследовано от общего описания для всех тэгов <INPUT>).
	2 пробела.
<input type="submit" value="OK" class="zagolovok"> 	Здесь, к описанию стиля, унаследованному от всех тэгов <INPUT>, добавляются описания стиля из класса zagolovok.
</form>	Конец формы.
<DIV ID="s1">	Начало слоя s1. Стиль слоя (его координаты и размеры) описан выше, в заголовке документа, в тэге <STYLE>.
<FONT color="blue">К тексту слоя 1 применен тэг FONT</FONT>	Данный текст будет выглядеть в соответствии со стилем тэга <FONT>, плюс в самом тэге <FONT> указан цвет шрифта – синий.
</DIV>	Конец слоя.
<DIV ID="s2" STYLE="position:absolute; left:350; top:200; width:300; height:100; visibility:show; zIndex:10">	Начало слоя s2. Координаты и размеры слоя, порядок наслоения, указан прямо в самом тэге <DIV>. Координаты не обязательно абсолютные. Допустимо, например, указание position:relative; top:10px; left:30px; – смещение на 10px вниз и на 30px вправо относительно нормальной позиции в документе. Можно вообще не указывать координаты – слой будет на своем обычном месте в документе. Помимо координат, для слоя можно указывать все те же свойства, что и для обычных стилей: шрифт, цвет, фоновая картинка слоя (background-image) и т.д.
<FONT color="blue"><SPAN class="zagolovok"> К тексту слоя 2 применен тэг FONT, однако он не действует из-за тэга SPAN </SPAN> </FONT>	Данный текст не будет выглядеть в соответствии со стилем тэга <FONT>, т.к. он находится внутри "контейнера" <SPAN> и его внешний вид однозначно определяется классом zagolovok. Никакие другие стили, кроме класса zagolovok, на текст внутри "контейнера" <SPAN> не влияют.
</DIV>	Конец слоя.
</BODY></HTML>	Конец документа.

Лекция 9. Язык VBScript

Язык VBScript – это несколько обрезанный Visual Basic (по соображениям безопасности отсутствуют возможности работы с файлами, вызова функций Windows API и др.), использующийся для написания небольших программок, встраиваемых в HTML-страницы, для придания им более привлекательного вида. Исходные тексты программ на VBScript записываются непосредственно в HTML-файл и выполняются web-браузером (в частности Internet Explorer-ом) при чтении и анализе файла. Программы на VBScript можно писать просто при помощи текстового редактора (только текст, без форматирования), редактируя HTML-файлы непосредственно или воспользоваться "Редактором сценариев" фирмы Microsoft (Microsoft Development Environment 6.0), который позволяет не только писать но и отлаживать программы на языках VBSript и JavaScript.

Программы на VBScript (также как и на JavaScript) должны заключаться в HTML-тэги <SCRIPT> </SCRIPT>. А чтобы текст программы не выводился на экран браузерами, которые не поддерживают VBScript, он дополнительно заключается в тэги комментария <!-- Программа -->. Пример см. ниже. Программы могут записываться в любом месте HTML-документа. Если программа не оформлена ключевыми словами SUB…END SUB, то она выполняется непосредственно при анализе страницы. Например, следующая программа выведет сообщение "Приветик" *после* того, как на HTML-странице будет отображен текст "До начала программы", но *до* того, как на HTML-странице будет отображен текст "После окончания программы".

```
<HTML>
<HEAD>  <TITLE>  Программа на VBScript   </TITLE>   </HEAD>
<BODY>
До начала программы
<SCRIPT LANGUAGE="VBScript">
<!--
MsgBox "Приветик"
-->
</SCRIPT>
После окончания программы
</BODY>
</HTML>
```

Если же текст программы оформлен ключевыми словами SUB…END SUB (FUNCTION…END FUNCTION), то такая программа может использоваться как обычная процедура (функция) вызываемая из других программ. Если имя процедуры удовлетворяет определенным требованиям, то такая процедура запустится при наступлении определенного *события*: нажатие кнопки, загрузка страницы, уход со страницы и т.д.

Синтаксис	Пример	Комментари
Sub Объект_Событие программа End Sub	Sub Knopka_onclick MsgBox "Меня нажали" End Sub	Если в HTML-документе существует кнопка (объект с именем knopka), то при нажатии на нее (событие onclick) сработает программа, которая выведет сообщение "Меня нажали".

Полный текст HTML-файла, реализующего приведенный в таблице пример, представлен ниже:

```
<html>  <head> <title> События !!! </title> </head>
<body>  <form method="POST" action="http://myself.com/something.asp">
<input type="button" value="Нажми меня" name="Knopka"> </form>
<script language=vbscript>
sub knopka_onclick
MsgBox "Меня нажали"
End Sub
--> </script>
</body> </html>
```

Процедуру обработки события можно называть и произвольным образом, но тогда название этой процедуры необходимо указать в качестве обработчика события в HTML – тэге элемента. Пример:

```
<HTML><BODY>
<SCRIPT LANGUAGE="VBScript">
sub info ( )
MsgBox "Вы перемещаетесь над гиперссылкой"
End Sub
</SCRIPT>
<a href="http://www.microsoft.com" onmouseover="info( )">  Гиперссылка </a>
</BODY> </HTML>
```

Из приведенных выше примеров можно сделать еще один вывод: основная мощь VBScript – не в самом языке, а в тех объектах, которыми он может манипулировать. Internet Explorer имеет ряд встроенных объектов (окно, документ, гиперссылки, формы и т.д.), каждый из которых имеет свой определенный набор свойств, методов и событий. Кратко объектная модель Internet Explorer приведена ниже:

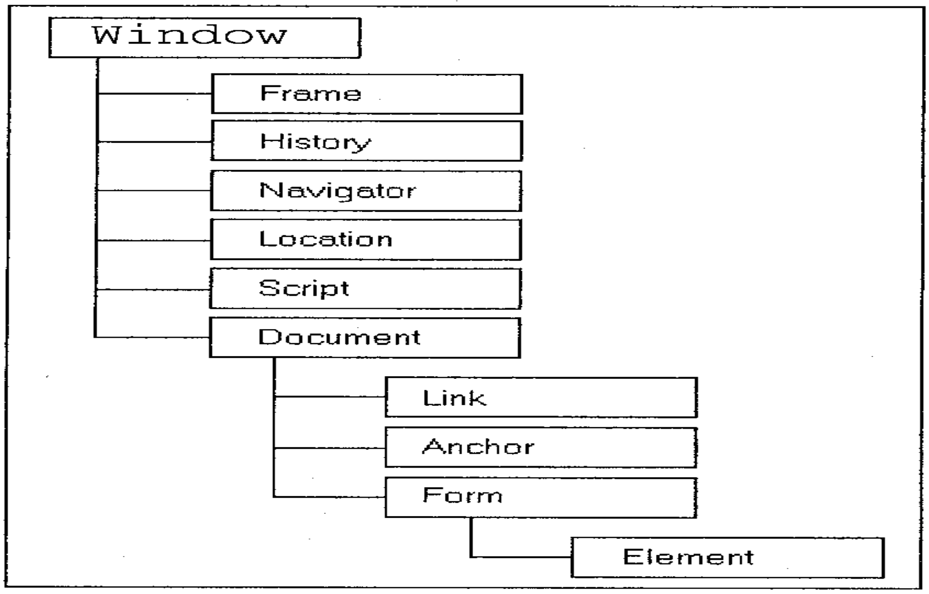


рис. Упрощенная объектная модель браузера Internet Explorer.

Приведенная на рисунке модель не являеется полной, однако даже из этой модели рассмотрим лишь некоторые обекты и лишь часть свойств этих объектов. Частично объектная модель будет также рассмотрена в лекциях по JavaScript.

Объект Window

Самый верхний объект в иерархии. При вызове свойств и методов, window указывать не обязательно.

Свойства

location	устанавливает или возвращает URL текущего окна. Можно загрузить в окно новую страницу, указав в теле документа: <SCRIPT LANGUAGE="VBSCRIPT"> location = "новый_url_адрес" </SCRIPT> или <SCRIPT LANGUAGE="VBSCRIPT"> window . location = "новый_url_адрес" </SCRIPT> Переход произойдет только после того, как будут обработаны все тэги <SCRIPT> на странице. Если в каком-то другом месте, вне функции (процедуры), также будет иметься строчка location = "url_адрес", то сработает последняя из строчек. Однако если URL-адрес был задан не статически в тексте скрипта, а введен пользователем с клавиатуры (см. метод Prompt), то объект location сработает немедленно.
status	устанавливает или возвращает текст, находящийся в строке состояния Internet Explorer. Пример: <SCRIPT LANGUAGE="VBSCRIPT"> status = "Приветик !" </SCRIPT> * В дальнейшем, для упрощения записи, в примерах не будут указываться тэги <SCRIPT> </SCRIPT>, хотя в реальном HTML-документе они, безусловно, необходимы.
defaultStatus	устанавливает или возвращает текст, отображаемый по умолчанию в строке состояния Internet Explorer. Например, URL текущего окна в строке состояния Internet Explorer можно вывести следующим образом: self.defaultStatus=location
self	возвращает текущий объект Window (ссылка на самого себя). Пример: см. выше.
name	возвращает имя текущего окна, если таковое определено. Пример: status=window.name
screenLeft	возвращает координату X окна, относительно левого края экрана. Пример: status=screenLeft
screenTop	возвращает координату Y окна, относительно верхнего края экрана. Пример: status = screenTop
screen . width	возвращает ширину окна. Пример: status = screen . width
screen .height	возвращает высоту окна. Пример: status = screen . height
sreen. availWidth	возвращает установленное в компьютере разрешение экрана по оси X. Пример: status = sreen . availWidth
sreen. availHeight	возвращает установленное в компьютере разрешение экрана по оси Y. Пример: status = sreen . availHeight
opener	возвращает окно, из которого открыто текущее окно.
parent	для страницы с фреймами, возвращает окно, находящееся на один уровень вложенности вверх, т.е. уровень, между тэгами <FRAMESET> которого содержится фрейм данного окна.
top	для страницы с фреймами, возвращает окно самого верхнего уровня вложенности.

Методы

alert	отображает простое окно с предупреждением и кнопкой **OK.** Пример: alert ("Приветик") Комментарий: можно использовать и MsgBox "Приветик", но alert – это метод объекта window, а MsgBox – команда VBScript
confirm	отображает окно сообщения, содержащее кнопки **OK** и **Cancel.** Пример: otvet=confirm("Ну и что нажмем ?") if otvet=true then alert ("Вы нажали OK")
prompt	предлагает пользователю ввести информацию. Аналогичен функции inputBox () в Visual Basic. Пример: otvet=prompt("Введите ФИО") alert ("Вы ввели " + otvet)
open	создает новое окно Internet Explorer и возвращает ссылку на него. Вот так создается "пустое" окно без меню, кнопок, полосы прокрутки, неизменяемое в размерах, которое, за исключением значка, ничем не напоминает Internet Explorer: set okno1=window.open("http://www.somewhere.com/file.htm", "imya", "height=100,width=100") Здесь imya – это имя окна, используемое в программах внутри этого окна (имя должно быть обязательно латинскими буквами). Для ссылок на окно извне необходимо использовать идентификатор okno1, созданный при открытии окна. Вот так можно закрыть окно: okno1.close Полный синтаксис метода приведен ниже: set идентификатор = window.open("URL-адрес", "имя окна", "параметры", заместить) , где заместить – принимает значения true (возврат через кнопку "назад" к открывшей странице невозможен, т.к. в истории посещений адрес исходной страницы замещается на адрес новой страницы) или false. параметры – строка параметров через запятую, заключенная в кавычки. Например: set okno1=window.open("file.htm", null, "height=100,width=100, copyhistory=yes, directories=no, location=no, menubar=no, resizable=no, scrollbars=no, status=no, toolbar=yes, fullscreen=no", true) Здесь: height – высота окна, width – ширина окна, copyhistory – переносить в окно "историю посещений" (кнопки "вперед" и "назад"), directories – панель ссылок "Microsoft" и др., location – поле ввода адреса, menubar – панель меню, resizable – размер окна можно изменять, scrollbars – полосы прокрутки, status – строка состояния, toolbar – панель кнопок, fullscreen – полноэкранный режим.
close	закрывает окно. Пример: window.close ( )
setTimeout	устанавливает таймер для срабатывания процедуры через определенное время и возвращает идентификатор таймера. Например, если необходимо запускать процедуру с именем info ( ) каждые 60 секунд, то соответствующий документ будет иметь вид: <HTML><BODY> <SCRIPT LANGUAGE="VBSCRIPT"> sub info( ) alert("Напоминаем каждую минуту") x=setTimeout("info()",60000) ' время указывается в милисекундах end sub info ( ) ' первый запуск info ( ) – при загрузке документа </SCRIPT> При помощи таймера чаще всего реализуется анимация (см. лекции по JavaScript).
clearTimeout	сбрасывает таймер с заданным идентификатором. Например: clearTimeout(x)
navigate	загружает в объект window заданный URL. Например, чтобы запросить у пользователя новый URL, а затем перейти по нему, можно воспользоваться кодом, приведенным ниже: x=prompt("Введите новый URL-адрес") navigate(x) Примечание: замечания по срабатыванию те же, что и для Location.
print	печать содержимого окна. Пример: window.print( )
moveTo	перемещает окно в координаты X,Y. Пример: y=window.moveTo (100,100)
moveBy	смещает координаты окна на величину X,Y. Пример: y=window.moveBy (10,10)
resizeTo	устанавливает ширину и высоту. Пример: y=window.resizeTo (800,600)
resizeBy	изменяет ширину и высоту окна на величину X,Y. Пример: y=window.resizeBy (10,10)
scrollTo	прокрутить окно до координат X,Y. Пример: y=window.scrollTo (100,100)
scrollBy	прокрутить окно на величину X,Y. Пример: y=window.scrollBy (100,100)
showModal Dialog	отображает диалоговое окно пользователя, созданное на основании html-страницы. Подробнее см. ниже в лекциях.

События

onload	Это событие происходит после окончания загрузки Web-страницы (включая и все рисунки) с сервера. При необходимости выполнить процедуру My_Program после полной загрузки страницы, нужно включить следующий тег: <BODY LANGUAGE="VBSCRIPT" ONLOAD="My_program">
onunload (выгрузка окна), onbeforeunload (перед выгрузкой окна), onresize (изменение размеров окна), onscroll (прокрутка окна), onbeforeprint (перед печатью), onafterprint (после печати), onfocus (при переключении на окно - получение фокуса), onblur (при уходе с окна – потере фокуса), onerror (при возникновении ошибки на странице), onhelp (при вызове справки – F1).	

Объект History

back	переход к предыдущему, ранее посещенному сайту. Метод аналогичен нажатию кнопки "Назад" на панели Internet Explorer. Пример: history.back( )
forward	переход к следующему, ранее посещенному сайту. Метод аналогичен нажатию кнопки "Вперед" на панели Internet Explorer. Пример: history.forward( )
go	переход на несколько, ранее посещенных сайтов вперед (+) или назад (-). Пример: history.go(-3)
length	число посещенных сайтов. Пример: status="Посетили "+history.length + " сайтов."

Объект Navigator

navigator.appName	название браузера. Пример: status= navigator.appName
navigator.appVersion	версия браузера. Пример: status= navigator.appVersion
navigator.platform	платформа (тип операционной системы), на которой исполняется браузер. Пример: status= navigator.platform
navigator.systemLanguage	язык системы (например, русский = ru). Пример: status= navigator.systemLanguage

Объект Location

Свойства

protocol	тип протокола, по которому загружена страница (http).
hostname	имя сервера, с которого загружена страница.
port	порт сервера, к которому производилось подключение (например 80, или 8080).
host	Комбинация имя сервера : порт.
pathname	полный путь до страницы (протокол, имя сервера, порт). Пример: status=location.pathname.

Методы

reload	перезагрузить страницу. Пример: location.reload (true) ' перезагрузить страницу с сервера location.reload (false) ' перезагрузить страницу из кэша
replace	заместить текущую страницу новой страницей. При этом теряется "история посещений сайтов" (history) и кнопки "вперед" и "назад" не работают. Пример: location.replace("file.htm")
assign	загрузка страницы. Пример: location.assign("http://www.microsoft.com/index.html")

Объект Document

Объект document представляет собой HTML-документ, загруженный в данный момент в объект window, и содержит все формы, элементы, ссылки, рисунки и компоненты ActiveX, существующие на странице. Доступ к свойствам (методам) документа осуществляется в формате document . свойство

Свойства

location	устанавливает/возвращает URL-адрес документа. Пример: alert (document . location)
title	возвращает заголовок документа, т.е. текст, расположенный между тэгами <TITLE>. Пример: alert (document . title)
linkColor	цвет гиперссылок в документе. Пример: document . linkColor = "gray"
alinkColor	цвет ссылок при удерживании кнопки мыши в нажатом состоянии. Пример: document . linkColor = "red"
vlinkColor	цвет посещенных ссылок в документ. Пример: document . linkColor = "green"
bgColor	цвет фона документа. Пример: document . bgcolor = "silver"
fgColor	основной цвет текста документа. Пример: document . bgcolor = "blue"
defaultCharset	кодировка документа по умолчанию. Аналогичен параметру charset тэга <**META**> в заголовке html-документа. Пример: document.defaultCharset="windows-1251"

readyState	определяет степень загрузки документа. Это свойство также определено для большинства подчиненных объектов объекта document (формы, рисунки), благодаря чему можно проверить степень их загрузки. Свойство может принимать следующие значения: uninitialized (инициализация/создание объекта еще не началось), loading(идет загрузка объекта), loaded (объект создан/проинициализирован), interactive(объект может взаимодействовать с пользователем, однако еще не все его данные получены), complete (загрузка объекта завершена). Ниже приведен пример, который проверяет загрузку документа через 10 секунд после начала. sub proverka( ) if document.readyState="complete" then alert("Документ полностью загружен") end sub x=window.setTimeout("proverka( )",10000)
selection	возвращает выделенный пользователем фрагмент html-документа. Пример: status = document.selection
lastModified	возвращает дату последнего обновления документа. Пример: status=document.lastModified
fileSize	возвращает размер файла документа. Пример: status=document. fileSize
fileCreateDate	возвращает дату создания документа. Пример: status=document. fileCreateDate
links	возвращает семейство гиперссылок в документе, отмеченных тэгами вида <A HREF = "http://adress.com" name="Metka"> </A>. Доступ к гиперссылке возможен по ее имени или позиции: alert(document . links (0)) ' вывод первой гиперссылки в документе alert(document . links ("Metka")) ' вывод гиперссылки с именем Metka alert(Metka) ' вывод гиперссылки с именем Metka y = document .links.length ' число гиперссылок в документе Гиперссылки имеют тот же набор событий, что и объект document (см. ниже). Однако эти события определены только относительно данной гиперссылки: нажатие клавиши обрабатывается только, если конкретная гиперссылка имеет фокус, событие onmouseover происходит только при перемещении над данной гиперссылкой и т.д. Пример: <HTML><BODY> <SCRIPT LANGUAGE="VBScript"> sub info ( ) alert ("Перемещаетесь над гиперссылкой") End Sub </SCRIPT> Гиперссылка </BODY> </HTML>
anchors	возвращает семейство закладок (якорей), т.е. позиций в документе, отмеченных тэгами <A NAME ="Metka"> </A> Всякая гиперссылка является закладкой, но не всякая закладка должна быть гиперссылкой. Доступ к закладкам – аналогичен доступу к гиперссылкам. alert(document . anchors (0)) ' вывод первой закладки в документе alert(document . anchors ("Metka")) ' вывод закладки с именем Metka alert(Metka) ' вывод закладки с именем Metka y = document .anchors.length ' число закладок в документе
images	возвращает семейство рисунков, содержащихся в документе. Доступ к рисункам возможен по их имени или позиции в документе. Пример: <IMG src="risunok.gif" name="ris1" alt="рисунок"> status=docunent.images(0).src ' вывод названия файла первого рисунка в документе status=docunent.images("ris1").src ' вывод названия файла рисунка с именем ris1 status=ris1.src ' упрощенная форма записи Каждый рисунок имеет следующие свойства, боьлшинство из которых доступны по чтению и записи: src (файл рисунка), lowsrc(файл, обычно небольшого рисунка низкого разрешения, который загружается браузером перед тем, как будет загружен основной, качественный рисунок), border (толщина рамки), heigth (высота в пикселях), width (ширина в пикселях), hspace и vspace (отсутп по горизонтали и вертикали), complete (только для чтения: true – рисунок загружен полностью, false – нет), name (только для чтения: имя рисунка в программе).
forms	возвращает семейство форм, содержащихся в документе. Доступ по номеру или по имени формы, аналогично доступу к рисункам. Пример: <FORM Name="forma1"> <INPUT type="text" name="fio1"> </FORM> document.forms(0).elements(0).value = "Иванов" document.forms("forma1").elements("fio1").value = "Иванов" forma1.fio1.value = "Иванов" Подробнее работа с формами будет рассмотрена ниже.

frames	возвращает семейство фреймов, описанных при помощи тэгов <FRAME>. Доступ по номеру или по имени фрейма, аналогично доступу к рисункам.
applets	возвращает семейство апплетов, описанных при помощи тэгов <APPLET>. Аппелет это независимая программа на языке Java, выполняющаяся в рамках html-страницы (подробнее см. лекции по Java). Доступ к апплету возможен по номеру его положения в документе или по имени, указанном в тэге <APPLET name=имя>, и аналогичен доступу к рисункам.
embeds	возвращает семейство объектов, внедренных в документ, например объектов ActiveX. Доступ по номеру или по имени объекта, аналогично доступу к рисункам.
plugins	возвращает семейство "плагинов" – специальных подключаемых к Internet Explorer программ-модулей, для отображения специфических данных. Например, файлов в формате pdf (Acrobat Reader) или анимации, созданной в программе Macromedia Flash. Доступ по номеру или по имени плагина, аналогично доступу к рисункам.
cookie	получает или устанавливает "ключик" для текущего документа. Cookie представляет собой подобие строковой переменной, которая записывается сервером на винчестере пользователя и используется каждый раз при заходе на данную web-страницу. В них сохраняется информация о пользователе, например список покупок, которые он сделал последний раз в Internet-магазине. Пример: status = document.cookie
referrer	возвращает URL документа с которого пользователь пришел на данную страницу.
parentWindow	возвращает объект window, в котором открыт данный документ.

Методы

write	записывает заданную строку в то место исходного текста html-документа, где расположен текст сценария. Например, если требуется курсивом вывести дату последнего редактирования документа, то в HTML-страницу надо вставить такой код: <SCRIPT LANGUAGE="VBSCRIPT"> document.write("<I> Этот документ был изменен: ") document .write(document . lastModified) document .write("</I>") </SCRIPT>
writeln	write, с переходом на новую строку. Важно отметить, что writeln (символ новой строки) игнорируется в HTML, если только он не помещен в теги <PRE> </PRE>.
open	открытие html-страницы с заданным адресом. См. метод open объекта window.
close	закрытие документа.
execCommand	Выполнение команды меню Internet Explorer. Пример: y=document.execCommand("print") Для примера приведем еще несколько команд: copy (копировать), paste(вставить), cut (вырезать), delete(удалить). Если предположить, что в документе имеется форма с именем forma1, содержащая два текстовых поля fio1 и fio2, в первое из которых пользователь ввел свое имя, то скопировать имя в поле fio2 можно следующим образом: forma1.fio1.select y=document.execCommand("copy") forma1.fio2.focus y=document.execCommand("paste")

События

Объект Document имеет следующие события:
onmouseover (перемещение мыши над документом), onmouseout (выход мыши за пределы элемента), onclick (щелчок мышью), ondblclick (двойной щелчок мышью), onmousedown (нажатие кнопки мыши), onmouseup (отпускание кнопки мыши), onkeypress (нажатие и отпускание клавиши), onkeydown (нажатие клавиши), onkeyup (отпускание клавиши), oncontextmenu (при вызове контекстного меню), onafterupdate, onbeforeupdate, oncellchange, onhelp, ondragstart, onselectstart, onstop, onpropertychange.

Пример:

```
<HTML><BODY onclick="info()">
<SCRIPT LANGUAGE="VBScript">
sub info ()
alert("Щелкнули по документу")
end sub
</SCRIPT>
Любой текст
</BODY> </HTML>
```

Объект Form

Объект Form представляет форму в HTML-документе. На форму можно сослаться из объекта Document, используя либо ее имя, либо ее индекс в массиве форм. Например:

```
<SCRIPT LANGUAGE="VBSCRIPT">
document . MyForm .FIO.value="Иванов"
document. forms(0). elements(0) .value="Иванов"
MyForm .FIO.value="Иванов"
</SCRIPT>
```

Форма описывается на языке HTML и содержит различные элементы: кнопки, текстовые поля, выпадающие списки и т.д. Объект Form предоставляет следующие свойства и методы для получения доступ к этим элементам:

Свойства

action	URL-адрес, куда отправляется форма.
encoding	тип кодировки формы. Обычно используется кодировка "text/html".
method	способ пересылки данных из формы на сервер: методы GET и POST (см. HTML).
target	имя окна, в котором будут отображаться данные формы. Так, при необходимости вывести форму с данными в другом окне Internet Explorer можно задать свойство Target или HTML-атрибут Target для формы.
elements	Возвращает семейство элементов формы (кнопки, текстовые поля и т.д.). Доступ к элементам возможен по их имени или позиции в форме. Например: <FORM action="" method="POST" name="MyForm"> <INPUT type="text" name="FIO"> </FORM> <SCRIPT LANGUAGE="VBSCRIPT"> MyForm . elements (0) . value = "Иванов" ' 1-му элементу формы присваиваем значение "Иванов" MyForm . FIO . value = "Иванов" ' элементу формы FIO присваиваем значение "Иванов" </SCRIPT>

Методы

submit	передает данные формы на сервер. Действие этого метода аналогично нажатию кнопки Submit (отправить). Пример: MyForm.submit( )
reset	очищает все поля формы. Действие этого метода аналогично нажатию кнопки Reset (сбросить). Пример: MyForm.reset( )
focus	передает фокус форме, активизируя ее. Пример: MyForm.focus( )
blur	форма теряет фокус. Пример: MyForm.blur( )
click	иммитирует щелчок мышью по форме. Пример: MyForm.click ( )

События

onsubmit	Возникает перед отправкой формы на сервер. Позволяет запустить программу, которая должна выполняться перед отправкой формы.
onmouseover (перемещение мыши над формой), onmouseout (выход мыши за пределы формы), onclick (щелчок мышью), ondblclick (двойной щелчок мышью), onmousedown (нажатие кнопки мыши), onmouseup (отпускание кнопки мыши), onkeypress (нажатие и отпускание клавиши, при наличии у формы фокуса), onkeydown (нажатие клавиши), onkeyup (отпускание клавиши), oncopy, oncut, onpaste, onbeforecopy, onbeforecut, onbeforepaste, onselectstart, oncellchange, oncontextmenu, onblur, onfocus, ondrag, ondrop, ondragstart, ondragenter, ondragleave, ondragover, onhelp, onreset, onsubmit, onscroll.	

Элементы формы

Как указывалось выше, обращение к элементам формы осуществляется либо по их индексу в семействе Elements, либо по их имени (см. пример). Каждый элемент HTML-формы имеет собственный набор свойств, методов и событий. Перечень событий элементов формы идентичен перечню событий формы (см. выше). Кроме того, элементы Password, Text, TextArea, Select имеют события: onselect (началось выделение текста) и onchange (содержимое элемента изменилось). Свойства и методы формы приведены в таблице.

Таблица

Свойства и методы отдельных элементов управления

<table>
<tr><th>Элементы</th><th>Уникальные свойства / методы</th><th>Общие свойства / методы</th></tr>
<tr><td>Submit,
Button,
Reset</td><td></td><td rowspan="5">1) Свойства:
form – возвращает объект Form, содержащий данный элемент.
name – имя элемента управления.
value – значение элемента управления. Например, текст введенный пользователем или надпись на кнопке.
disabled – при присвоении свойству значения true, отключает элемент управления.
2) Методы:
click – имитация щелчка мышью по элементу управления.
focus – получение фокуса элементом управления.
blur – потеря фокуса элементом управления.</td></tr>
<tr><td>Chekbox,
Radio</td><td>checked – возвращает TRUE, если флажок установлен.
defaultChecked – флажок установлен по умолчанию (TRUE).</td></tr>
<tr><td>Text,
Password</td><td>size – длина текстового поля (символов).</td></tr>
<tr><td>Textarea</td><td>rows – число строк, cols – число столбцов.
select – выделение текста.</td></tr>
<tr><td>Select
(список)</td><td>size – число одновременно отображаемых элементов списка.
length – число элементов списка.
multiple – возможность выбора одновременно нескольких элементов списка.
selectedIndex – номер выбранный элемент списка.
item – доступ к элементу списка по имени или номеру.</td></tr>
</table>

Пример:

```
<HTML><BODY> <SCRIPT LANGUAGE="VBScript">
sub info ( )
otvet1="Вы ввели текст: "+ forma1.vvod.value
alert(otvet1)
end sub
</SCRIPT>
<FORM name="forma1">
<INPUT type="text" name="vvod">
<INPUT type="button" name="knopka" value="Проверить" onclick="info">
</FORM>
</BODY> </HTML>
```

Создание диалогов пользователя (метод showModalDialog объекта window)

Метод showModalDialog объекта window отображает html-страницу, как модальный диалог (пока диалог не будет закрыт, работать с открывшей его страницей нельзя). Метод showModalDialog позволяет использовать не только встроенные диалоговые окна типа alert и prompt, но и создавать собственные диалоговые окна пользователя. Для этого сначала создается html-документ диалога, а затем этот документ вызывается в качестве диалогового окна. Пример:

Окно диалога – файл modal.htm :

```
<HTML> <BODY text="red" bgcolor="Silver">
Это окно диалога.
<form name="forma1">
<input type="button" name="knopka1" value="OK" onclick="say1( )">
<input type="button" name="knopka2" value="Cancel" onclick="say2( )">
</form>
<SCRIPT language="vbscript">
args=window.dialogArguments  ' считывает массив аргументов переданных диалогу
forma1.knopka1.value=args(0)   ' меняет надпись на кнопке1, в соответствии с первым аргументом
forma1.knopka2.value=args(1)   ' меняет надпись на кнопке2, в соответствии со вторым аргументом

sub say1( )                     ' процедура срабатывает при нажатии кнопки 1 (см. тэг INPUT)
window.returnValue= forma1.knopka1.value   ' возвращает из диалога значение нажатой кнопки
window.close( )                 ' закрывает диалог
end sub

sub say2( )                     ' процедура срабатывает при нажатии кнопки 2
window.returnValue= forma1.knopka2.value
window.close()
end sub
</SCRIPT></BODY></HTML>
```

В приведенном выше диалоге созданы две кнопки, надписи на коорых задаются в качестве аргументов диалога. При щелчке по кнопкам (событие onclick) вызываются соответствующие процедуры (say1 или say2), которые возвращают (window.returnValue) значение нажатой кнопки и закрывают диалог (window.close). Диалог вызывается из документа приведенного ниже:

Документ из которого вызывается диалоговое окно:

```
<HTML><BODY> <SCRIPT LANGUAGE="VBScript">
args = Array("OK", "Отмена")                    ' для JavaScript было бы args = new Array("OK", "Отмена");
otvet=window.showModalDialog("zmodal.htm",args, "font-size:20; dialogWidth:10; dialogHeight:7; dialogTop:10; dialogLeft:10; center:no")
alert(otvet)
</SCRIPT> </BODY> </HTML>
```

Здесь в переменную otvet попадает результат работы диалога, а сам диалог вызывается при помощи метода showModalDialog, параметры которого обозначают следующее:

"modal.htm" – адрес html-страницы, используемой в качестве диалогового окна.

args – аргументы, передаваемые диалогу. Если аргументов нет – можно указать null.

"параметры"– строка параметров font-size(размер шрифта), dialogWidth (ширина диалога в процентах экрана), dialogHeight (высота диалога в процента экрана), dialogTop(координата Y в пикселах от верха экрана), dialogLeft (координата X в пикселах от левой стороны экрана), center (размещение диалога по центру экрана).

Цикл For..Next в VBScript.

Особенностью VBScript (в отличие от VBA) являеется нестандартное написание цикла For .. Next, без указания в команде Next переменной – счетчика цикла. Пример:

```
<HTML> <BODY> <SCRIPT LANGUAGE="vbscript">
for i = 1 to 2
MsgBox "Сообщение выводится 2 раза"
Next
</SCRIPT> </BODY> </HTML>
```

Лекция 10. Объекты ActiveX - технология корпорации Microsoft

Язык HTML – это просто язык гипертекстовой разметки, который определяет, как будет выглядеть страница, но не может придать странице динамичности и интерактивности. Для решения этой проблемы, фирмой Sun были предложены Java-апплеты (небольшие программы, которые вставляются непосредственно в HTML-страницу и выполняются при помощи броузера). В ответ на эту инициативу, фирма Microsoft создала альтернативную технологию - объекты ActiveX, которые также представляют собой небольшие программы, вставляемые в HTML страницу и выполняемые броузером. Объект может представлять собой как отдельную кнопку или другой элемент управления, так и целую программу. Как и любой объект, каждый объект ActiveX имеет свойства, методы и события, которыми можно манипулировать из программ на VBScript и в JavaScript в виде "ИмяОбъекта.Свойство". Таким образом, созданные программистом объекты ActiveX, становятся как бы продолжением встроенных объектов броузера.

Загрузка объекта ActiveX происходит следующим образом: если объект ActiveX ранее не использовался, то он загружается через сеть и устанавливается на компьютере пользователя, регистрируясь в реестре так, чтобы при повторной загрузке этой страницы (или любой другой страницы, содержащей такой же объект), его не надо было бы загружать по сети. Это, по мнению Microsoft, является преимуществом объектов ActiveX перед апплетами Java, т.к. объекты ActiveX загружаются по сети лишь один раз. Другим "преимуществом" объектов ActiveX является то, что они имеют полный доступ к ресурсам компьютера (для сравнения: апплеты Java выполняются "в песочнице" под контролем менеджера безопасности и даже не имеют возможности записывать/читать данные на винчестер). Эти "преимущества" объектов ActiveX означают следующее: пользователь, заходя на Web-страницы, загружает и устанавливает у себя на компьютере программное обеспечение неизвестного происхождения и назначения, которое может сделать с компьютером под управлением Windows 9x все что угодно. Таким образом, налицо проблема с безопасностью объектов ActiveX, которые могут быть использованы для нанесения ущерба (вирусы) и кражи информации (троянские программы). Для решения этой проблемы Microsoft предлагает схему, напоминающую использование нотариусов: разработчик программного обеспечения предоставляет центру сертификации сведения о себе и обязательство не создавать программы, наносящие вред клиентам. Центр сертификации (солидная уважаемая организация, аналог нотариальной конторы), выдает разработчику сертификат ограниченного срока действия (аналог печати предприятия), который позволит разработчику подписывать создаваемые им программы электронной цифровой подписью. Электронная цифровая подпись позволяет от имени центра сертификации гарантировать:

1) Данная программа распространяется именно этим автором (т.к. только у него есть сертификат - печать) и, следовательно, он будет нести ответственность за потенциально возможный ущерб.
2) Программа не была изменена злоумышленником (изменение даже одного бита программы сделает подпись некорректной).

Перед загрузкой и установкой объекта ActiveX броузер выведет сведения об авторе и центре сертификации. Если вы не доверяете человеку, который создал объект, или не доверяете репутации центра сертификации (автор, в частности, может и сам себе выдать сертификат), то можете отменить загрузку объекта. Настройками в броузере можно вообще отказаться от использования объектов ActiveX.

Объект ActiveX вставляется в текст HTML-страницы при помощи тэга <Object> - для броузера Internet Explorer. В броузере Netscape используется тэг <Embed>, т.к. объекты ActiveX поддерживаются не непосредственно, а через механизм подключаемых модулей "Plugins". Как и все сложные HTML-теги, тег <Object> обладает обширным набором различных атрибутов. Пример:

```
<OBJECT  ID="timer"  CLASSID="clsid:59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
CODEBASE= "http://sait.com" ALIGN="middle" WIDTH="116px"  HEIGHT="50px">
<PARAM  NAME="FontName"  VALUE="Times">
<PARAM  NAME="FontHeight"  VALUE="16">
</OBJECT>
```

Тег <Object> имеет следующие атрибуты.

1) ID — определяет имя объекта, которое будет использовано при обращении к последнему из сценариев на VBScript и JavaScript.

2) ClassID — при добавлении в систему компонента ActiveX информация о нем должна быть занесена в реестр Windows. При этом ему присваивается уникальный идентификатор (GUID — global unique identifier), который используется для создания экземпляра соответствующего класса ActiveX-компонента. Получив информацию о компоненте ActiveX, внедренном в Web-страницу, броузер в первую очередь использует атрибут ClassID для обращение к реестру Windows (API-функция CoGetclassObject). Если компонент инсталлирован и зарегистрирован, броузер применяет атрибут ClassID для создания экземпляра компонента ActiveX. В случае когда для данного атрибута не находится соответствующего GUID из реестра Windows, компонент загружается через Internet (API-функция CoGetclassObjectFromURL).

3) CodeBase — как указывалось выше, если компонент не инсталлирован в системе клиента, броузер должен загрузить и инсталлировать его. Поэтому при включении компонента ActiveX в Web-страницу необходимо

задать и значение атрибута CodeBase (то есть URL), по которому броузер может найти и загрузить элемент управления.
4) Align — определяет способ размещения объекта на Web-странице. Возможные следующие значения этого атрибута:

Значение	Действие
Baseline	Устанавливает нижнюю границу объекта на уровне базовой линии обтекающего его текста
Center	Задает горизонтальное выравнивание объекта по центру страницы
Left	Выравнивает объект по левому краю страницы (при этом текст обтекает объект справа)
Middle	Середина объекта располагается на уровне базовой линии охватывающего его текста
Right	Объект выравнивается по правому краю страницы
Text Bottom	Нижняя граница объекта выравнивается по нижней границе текста
TextMiddle	Середина объекта располагается на одном уровне с серединой обтекающего текста
TextTop	Верхняя граница объекта выравнивается по верхней границе обтекающего текста

5) Width — задает ширину объекта при отображении его на Web-странице.
6) Height — применяется для определения высоты объекта при отображении его на Web-странице.

Возможны также следующие параметры:
7) Border — определяет толщину рамки вокруг объекта.
8) Codetype — используется для проверки совместимости приложения броузера с объектом, загружаемым в него для просмотра.
9) Data — применяется для определения файла, содержащего необходимую для объекта информацию. Например, если создается экземпляр элемента управления Multimedia, то свойство Data будет содержать ссылку на AVI-файл (видеоданные).
10) Declare — указывает, что необходимо, не создавая экземпляр объекта, объявить только его класс в контексте страницы. Это свойство используется при последующем создании в документе перекрестных ссылок на объект или при использовании объекта в качестве параметра для другого объекта в VBScript.
11) Hspace — определяет размеры отступов справа и слева от границ видимой области объекта.
12) Name — если объект расположен в блоке <Form>...</Form>, то при наличии данного атрибута он будет передаваться на сервер с применением определенного для данной формы HTTP-метода. Благодаря этому атрибуту в формах вместо элементов управления HTML можно использовать элементы управления ActiveX.
13) Shapes — задает определенные области внедренного объекта как гиперссылки.
14) Standby — определяет текст, который будет появляться на Web-странице при загрузке или создании экземпляра объекта.
15) Type — задает тип кодировки информации.
16) Vspace — задает размеры отступов от верхней и нижней границ области отображения объекта.

Применяемые внутри тегов <Object> </ Object >, тэги <PARAM> позволяют задать свойства объекта, экземпляр которого создается в HTML-документе. Так в указанном выше примере при помощи тэга <PARAM NAME="FontName" VALUE="Times"> устанавливается свойство FontName создаваемого объекта, равным Times.

Лекция 11. Язык Java

Java - язык, разработанный Sun Microsystems изначально для приложений бытовой электроники и позднее перенесенный в Internet, где он и стал одним из основных языков программирования. Java позволяет создавать четыре типа приложений:

- Приложения командной строки (выполняются из командной строки, как в DOS или Unix).
- Приложения с графическим интерфейсом (GUI-приложения, как в Windows).
- Пакеты (библиотеки классов, в чем-то аналогичны DLL-файлам).
- Апплеты (мини-программы, которые не могут выполняться самостоятельно, а выполняются в среде интернет -браузера, типа Internet Explorer или Netscape Navigator, и используются для придания web-страницам привлекательного вида).

1. Java — интерпретируемый язык (виртуальная Java-машина)

Java — интерпретируемый язык. Для этого есть свои причины. Основная проблема программ, которые предполагается использовать в Интернет – это необходимость выполнять программу на различных типах компьютеров. В Java эта проблема решена, для этого между компилятором и компьютером вводится посредник, называемый *виртуальная Java-машина (Java Virtual Machine, JVM).* Java-компилятор транслирует исходный текст программы в низкоуровневые байт-коды, которые не зависят от типа компьютера и одинаковы для всех JVM-машин. Байт-коды записываются в файл класса, с расширением ".class". Каждая команда байт-кода состоит из *однобайтного кода операции* и одного или нескольких аргументов. В процессе выполнения программы JVM-машина читает и интерпретирует эти байт-коды, в результате чего Java-программа может выполняться на любом типе компьютера, для которого написана JVM-машина (Windows, Unix, Macintosh и т.д.). Все Web-браузеры, выполняющие Java-апплеты, также имеют встроенную JVM-машину. Такой подход удобен как для разработчиков, т.к. нет необходимости переписывать Java-программу для каждого из существующих типов платформ компьютеров. Единственным недостатком JVM-машины является то, что интерпретация байт-кода выполняется намного медленнее, чем работает скомпилированная программа на машинно-зависимом языке. Для преодоления этой проблемы JVM-машина дополняется ЛТ-компилятором (Just-in-Time) который читает байт-код и транслирует его в машинно-зависимые команды. Таким образом, например, нет необходимости 100 раз интерпретировать одни и те же команды цикла, т.к. они уже скомпилированы в машинно-зависимый код. В результате Java-программа может работать почти с той же скоростью, что и машинно-зависимая программа на C++.

2. Java — объектно-ориентированный язык

Язык Java является объектно-ориентированным и имеет много общего с языком C++. Как и любой объектно-ориентированный язык, программа Java строится на иерархии объектов, которые представляют из себя единство данных (свойств) и процедур работы с этими данными (методов). Для упрощения языка, в Java не все данные являются объектами. Например, булевские типы, числа и другие простые типы данных не есть объекты, хотя в языке все-таки имеются упакованные объекты для этих типов данных. В остальном, Java — строгий объектно-ориентированный язык: никакая объявляемая переменная или процедура не может не входить в состав какого-либо объекта. Описание свойств и методов объекта называется классом. Объекты создаются на основании классов. Классы могут наследовать свойства и методы других классов, например: "Все люди (класс) имеют возраст (свойство), футболисты – это порожденный класс от класса люди и он также наследует свойство возраст". В дополнение к наследованию, в порожденных объектах можно добавлять новые свойства и методы, а также модифицировать унаследованные, подменяя их своими методами и свойствами. Можно создавать абстрактные классы – классы в которых объявлены методы, но не описана их реализация (хотя бы одного). Задача описания реализации метода ложится на классы, порожденные от абстрактного, т.к. объект нельзя создать пока полностью не описаны все его свойства и методы. Java поддерживает только простое наследование, т.е. каждый класс в отдельный момент времени может порождаться только от одного какого-либо класса (а не от нескольких, как при множественном наследовании). При таком подходе к наследованию устраняются проблемы с классом, порожденным от противоречивых или взаимоисключающих классов. В компенсацию за отсутствие множественного наследования в Java можно создавать совершенно абстрактные классы, называемые *интерфейсами (interface),*которые позволяют описывать методы, разделяемые между несколькими классами. Интерфейсы могут содержать только свойства-константы и объявления методов, без описания их реализации. Объекты могут порождаться от любого количества интерфейсов. К примеру, существует интерфейс "Спортсмен" и интерфейс "Военный", каждый из которых определяет свой специфический набор свойств и методов. Каждый человек (объект) может быть объявлен одновременно и "Спортсменом" и "Военным", получив свойства и методы этих двух интерфейсов.

3. Апплеты Java

Апплет — это встроенная в Web-страницу мини-программа на языке Java, которая используются для придания web-странице привлекательного вида. Апплет не может выполняться самостоятельно, для этого нужен Java-совместимый Web-браузер. Браузер загружает HTML-документ, одновременно с ним загружается и выполняется Java-апплет. Цикл загрузки апплетов приведен ниже:

1. Загружается HTML-файл.
2. Обнаруживается тег <APPLET>.
3. Файл класса (файл с программой), указанного в APPLET, загружается с сервера.
4. Распознаются и загружаются классы, на которые ссылается класс APPLET.
5. Класс APPLET вызывает методы init () and start ().
6. Апплет выполняется (отображается в окне браузера или вне его, если апплет использует собственный кадр *—frame).*

4. Встраивание апплетов в HTML-страницы

Апплет – часть HTML-страницы, и встраивается в нее через тег <applet>, например:

<APPLET
CODEBASE="http://www.spravka.by"
CODE="telefon.Class"
WIDTH="100px" HEIGHT="100px" ALT="Альтернативный текст" NAME="TelKniga" ALIGN="middle"
VSPACE="10" HSPACE="10">
<PARAM NAME="Adres" VALUE="Минск">
<PARAM NAME="Telefon" VALUE="5550137">
Если вы видите этот текст, значит ваш браузер не поддерживает апплеты! Пора сменить…
</APPLET>

Ниже кратко описаны все показанные атрибуты:

Обязательные атрибуты	Допустимые значения
CODE	Допустимое имя файла класса апплета
WIDTH	Ширина апплета в пикселах
HEIGHT	Высота апплета в пикселах
CODEBASE	Допустимая URL-ссылка на каталог, в котором располагаются файлы класса этого апплета
ALT	Альтернативный текст для случая, когда Java-совместимый браузер не может выполнить апплет
NAME	Частное имя апплета, по которому другие апплеты, расположенные на той же HTML-странице, могут обращаться к нему
ALIGN	Выравнивание апплета; допустимые значения: left, right, top, texttop, middle, absmiddle, baseline, bottom, absbottom
VSPACE	Пустые места (задаваемые в пикселах) сверху и снизу от апплета
HSPACE	Пустые места (задаваемые в пикселах) справа и слева от апплета
PARAM	Параметры, передаваемые апплету

5. Безопасность Java апплетов

Неприятно сознавать, что Web-страницы, которыми переполнен Интернет, могут содержать чужие программы, которые выполняются на вашем компьютере. Истина, однако, заключается в том, что Java-апплеты — безопасный способ распространения программ через Internet. Это объясняется тем, что интерпретатор Java не запустит апплет до тех пор, пока не убедится в том, что байт-код апплета безопасен (систему безопасности Java см. ниже). Более того, Java-апплеты не только защищены системой безопасности, но и сам язык Java построен так, что апплеты практически не в состоянии повредить систему. На апплеты, загружаемые по сети, накладываются следующие ограничения:

- Апплет не может читать или модифицировать файлы локальной файловой системы, создавать, переименовывать или копировать файлы и каталоги.
- Апплет не может создавать произвольные сетевые соединения, за исключением связей с той хост-машиной, с которой апплеты были считаны.
- Апплет не может вызывать внешние программы посредством таких системных вызовов, как fork или exec, загружать в клиентской машине динамические библиотеки.
- Апплет не может манипулировать какими-либо группами Java-потоков, за исключением их собственной группы потоков, порожденной из главного потока апплета.

- Апплет не может останавливать работу виртуальной Java-машины, игнорировать или подменять проверку системы безопасности.
- Доступ апплета к информации о системе ограничен (недоступна информация: домашний и текущий каталог пользователя, регистрационное имя пользователя, каталог инсталляции Java, путь к Java-классам и др.).

6. Система безопасности Java

Важным достоинством Java-приложений является защищенность. Во-первых, сам язык способствует написанию более защищенных и устойчивых к сбоям программ. Во-вторых, язык имеет встроенную систему безопасности. К особенностям языка, заставляющим писать безопасный код относятся :

1) **Строгая ориентация на объекты.**
   Технология объектно-ориентированного программирования значительно снижает вероятность возникновения ошибок в программах, из-за подхода к объекту как к завершенной автономной единице. Это исключает возникновение таких трудно обнаруживаемых ошибок, как ошибки, возникающие при взаимодействии двух по отдельности нормально работающих процедур, но ссылающихся на одну и ту же область памяти, в результате чего последствия их совместной работы не предсказуемы. Чем меньше таких ошибок – тем устойчивее работа программы и меньше возможности проникновения в систему из-за сбоев в ней.
2) **Строгая типизация и безопасное преобразование типов**.
   В целях безопасности *преобразования типов* проверяются как статически, так и динамически; это гарантирует то, что объявленный на этапе компиляции тип объекта будет точно соответствовать типу объекта во время выполнения, даже если по отношению к этому объекту выполнялись операции преобразования типов. Контроль за преобразованием типов препятствует преднамеренной подмене типов данных.
3) **Отсутствие указателей.**
   Все элементы данных всегда имеют имя. Каждая простая структура данных или фрагмент кода имеют идентификатор, позволяющий полностью контролировать их. Нельзя сослаться на область памяти (прочитать/записать), не относящейся к переменным Java-программы. Если бы указатели существовали – это дало бы возможность писать (читать) в любую область памяти компьютера, несанкционированно подменять (получать) данные, выполнять свой код.
4) **Позднее распределение памяти и связывание.**
   Позднее связывание гарантирует, что точное местоположение ресурсов на этапе выполнения происходит в самый последний момент. Позднее связывание представляет серьезное препятствие на пути внешних атак благодаря специальным соглашениям, касающимся выделения памяти для этих ресурсов.
5) **Автоматическая сборка мусора и неявное управление памятью.**
   В языках C и C++ программист явным образом распределяет память, освобождает ее и следит за всеми указателями на выделенную память. Зачастую это усложняет программы и является главным источником ошибок и уязвимости к атакам типа "переполнение буфера" (злоумышленник, передав программе слишком большой, не предусмотренный программистом, параметр может вызвать крушение системы или выполнение своего кода), нелегальное копирование, захват полномочий. В языке Java программист не должен явно управлять памятью. Выделение и освобождение памяти выполняются автоматически и корректно. При освобождении неиспользуемой памяти применяется механизм сборки мусора (дефрагментация памяти).
6) **Классы и методы final.**
   Классы и методы можно объявлять как final, запрещая тем самым создание подклассов и переопределение методов. Такое объявление препятствует злонамеренному изменению проверенного кода.
7) **Проверка имен.**
   Классы Java описываются внутри *пакетов.* Имена классов связаны с названиями пакетов. Пакеты гарантируют, что код, полученный из сети, отличается от локального кода. Принятая библиотека классов не может по ошибке или преднамеренно заменить локальные библиотеки проверенных классов или перехватить их права, даже если эти библиотеки имеют одинаковые имена. Это также защищает от непроверенных, случайных взаимодействий локального и принятого классов.
8) **Синтаксические конструкции для защищенных потоковых структур данных.**
   Java является многопотоковым языком и обеспечивает защищенный потоковый доступ к структурам данных и объектам.
9) **Уникальные манипуляторы для объектов**.
   С каждым объектом в языке Java связан уникальный хэш-код (hashcode). Это означает, что в любой момент возможен мониторинг состояния Java-программы.
10) **Безопасность на уровне компиляции Java-кода.**
   Во время компиляции анализируются все механизмы защиты, существующие в синтаксисе языка Java, включая проверку согласованности объявлений private и public, правильности типов и инициализации всех переменных в соответствии с предопределенными значениями.

Как уже говорилось, помимо особенностей языка Java, "заставляющих" писать безопасный код, в Java имеется и встроенная система безопасности, которая состоит из следующих элементов:

- **ClassLoader** (загрузчик классов)
- **Verifier** (верификатор)
- **SecurityManager** (менеджер безопасности)

Эта модель известна под названием *sandbox* (песочница). Все Java-приложения, загруженные из сети, выполняются "в песочнице", т.е. не имеют полного доступа к ресурсам компьютера (см. "Безопасность Java-апплетов") и контролируются системой безопасности Java.

Загрузчик классов определяет, когда и каким образом классы могут быть добавлены в работающую систему и защищает целостность системы, например запрещает загрузку поддельного менеджера безопасности. Он выполняют две основные функции:

- собственно загрузку байт-кода (с локального диска, по сети, из области памяти)
- определение пространства имен (*namespaces*) для различных классов и способов их взаимодействия (отделяя, к примеру, локальные классы от загруженных по сети).

Есть два вида загрузчиков - первичный (*primordial*) и реализованный в виде объекта (*Class Loader Object*). Первичный существует в единственном экземпляре и является составной частью виртуальной машины. Он занимается загрузкой доверенных классов (обычно находящихся на локальном диске). Загрузчик второго типа представляет собой обычный Java-объект. С его помощью можно осуществить загрузку класса по сети либо динамическое создание класса программой. Алгоритм действий загрузчика обычно выглядит так:

1. Определить, не был ли загружен этот класс раньше, и, если да, вернуть его.
2. Проконсультироваться с первичным загрузчиком на предмет существования внутреннего класса с этим именем (во избежание подмены внутренних классов Java классами загружаемыми по сети).
3. Запросить у менеджера безопасности разрешение на загрузку данного класса.
4. Считать файл класса в виде массива байтов - по сети, с диска и т. п.
5. Создать экземпляр класса Class.
6. Загрузить другие классы, используемые данным.
7. Передать класс верификатору на проверку.

Второй рубеж обороны - **верификатор**, проверяющий загружаемый байт-код на корректность, так как у нас нет никакой гарантии, что загружаемый код был получен в результате работы компилятора Java, а не подправлен вручную или не сгенерирован специальным «враждебным» компилятором. После того как код прошел верификацию, гарантируется, что файл класса имеет корректный формат, параметры всех операций имеют правильный тип, в коде не происходит некорректных преобразований типов (например, целого числа в указатель), нет нарушений доступа, нет переполнения стека и т. п. Таким образом, проверяется все, что только можно проверить до начала исполнения программы. Верификатор встроен в виртуальную машину и недоступен из Java-программы.

Класс **SecurityManager** (менеджер безопасности) отвечает за политику безопасности приложения. Он позволяет приложению перед выполнением потенциально опасной операции выяснить, выполняется ли она классом, загруженным первичным загрузчиком, либо с помощью некоторого ClassLoader по сети (к последнему доверия должно быть гораздо меньше). Далее менеджер безопасности может определить, разрешить ли эту операцию или запретить. Контролируется работа с файлами (создание, удаление, чтение, запись), запуск программ, подключение библиотек DLL, создание входящих и исходящих сетевых соединений, подмена менеджера безопасности и загрузчика классов, обращение к системным ресурсам, к ресурсам виртуальной машины, защита потоков и групп потоков друг от друга и т.д. Для каждого кода (апплета, Java-программы) имеется возможность установить права доступ и указать, что код, обладающий определенными правами доступа, имеет право на определенные действия (например, создание сетевого соединения с другой машиной). Никакой Java-код не считается безопасным по умолчанию. Локальный код может быть подвержен тем же проверкам, что и код апплета, хотя, конечно, никто не мешает ослабить этот контроль с помощью настроек. В менеджере безопасности имеется также возможность цифровой подписи классов (аналог сертификатов в ActiveX). На классы подписанные солидной организацией можно не накладывать стандартные ограничения. В целом, система безопасности (версии JDK 1.1 и 1.2.) представлены ниже:

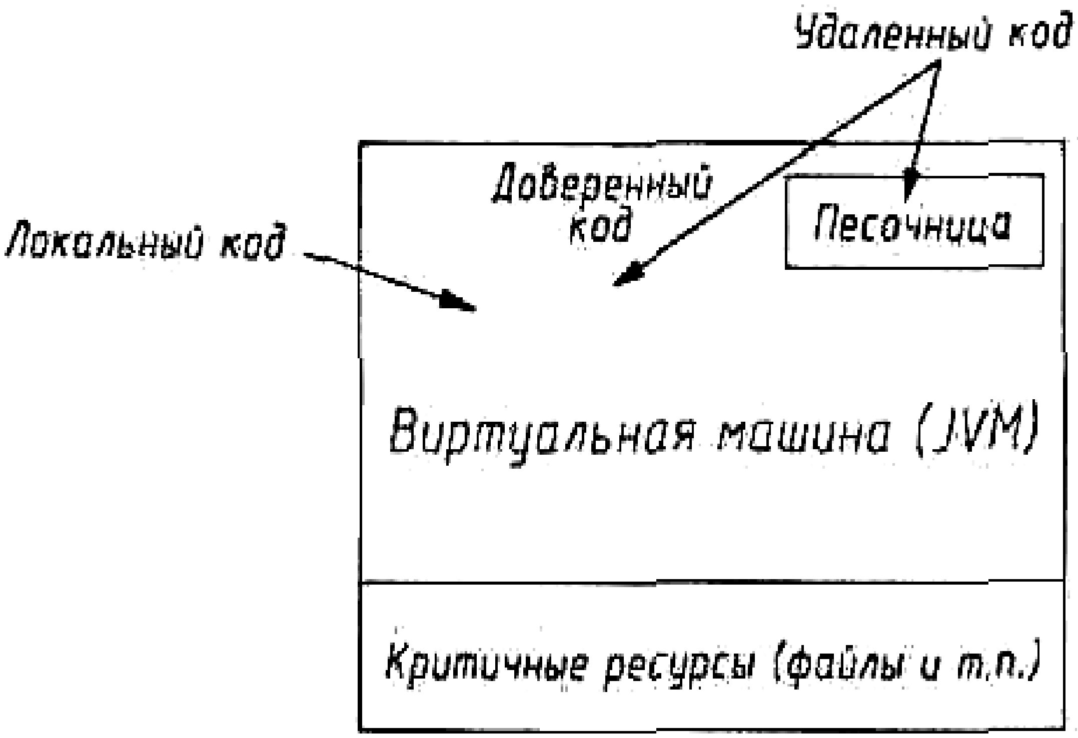


Рис. 10.2. Модель безопасности JDK 1.1

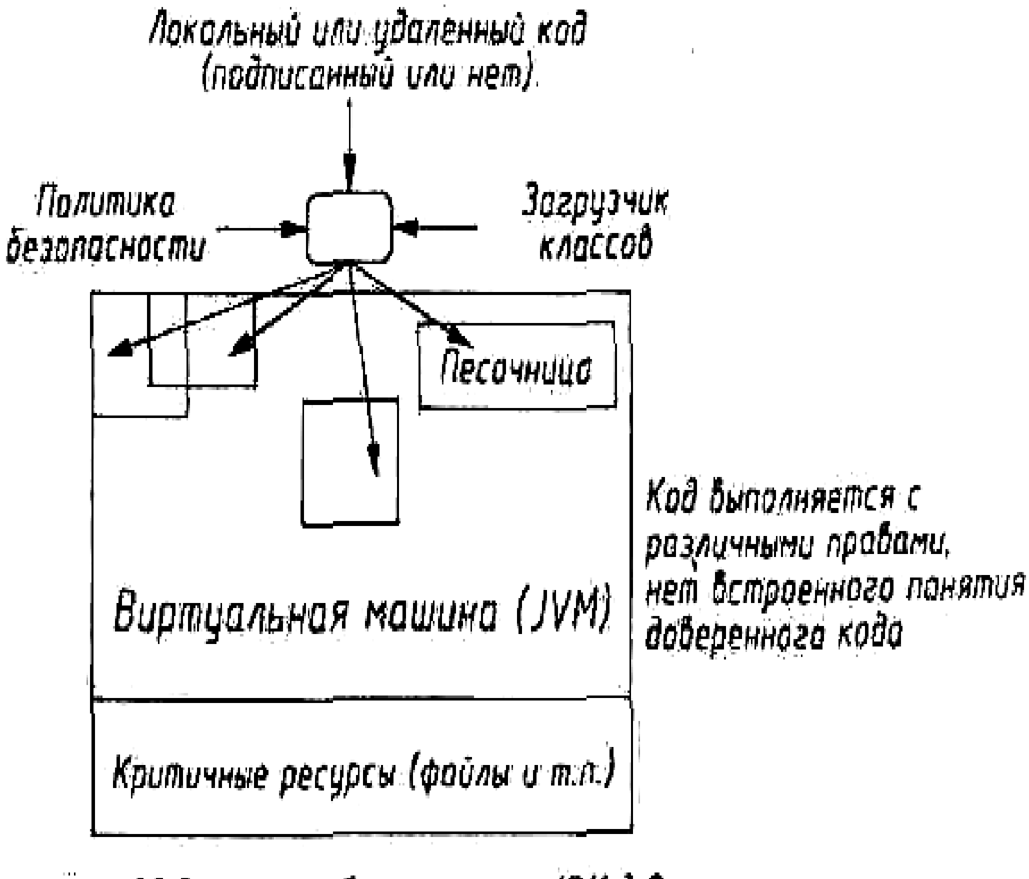


Рис. 10.3. Модель безопасности JDK 1.2

7. Интерфейс Java API

Java Application Programming Interface (Интерфейс прикладного программирования), или *Java API,* — это набор классов, разработанных компанией Sun для работы с языком Java. Этот интерфейс помогает при создании собственных классов, апплетов и приложений. Используя уже готовые классы, можно написать Java-приложение длиной всего в несколько строк в отличие от сотен программных строк, необходимых для создания программы на С. Классы в Java API сгруппированы в пакеты, в которых могут быть по несколько классов и интерфейсов. Более того, каждый элемент может также иметь различные свойства, например, поля и/или методы. Ниже перечислены некоторые имеющиеся или разрабатываемые API-интерфейсы:

Стандартный Java API

java.io	Пакет java.io служит в языке Java стандартной библиотекой ввода/вывода.
java.util	Пакет java.util главным образом состоит из различных полезных классов, которые трудно отнести к какому-либо другому пакету, например класс Date, облегчающий работу с датами, класс Hashtable, класс stack и др.
java.net	пакет Java.net предоставляет средства для связи с удаленными ресурсами, для чего можно создавать сокеты, подключаться к ним или использовать URL-ссылки. К примеру, при помощи этого пакета можно создать собственные клиентские и/или серверные программы для протоколов Telnet, Chat или FTP.
java. awt	Пакет java.awt - оконный пользовательский интерфейс (Abstract Window Toolkit, AWT). В нем содержатся средства, позволяющие создавать мощные, привлекательные и удобные графические оконные интерфейсы для апплетов и автономных программ. В этом пакете имеются интерактивные средства, например, Button и TextFieid, а также класс Graphics, предоставляющий средства для рисования фигур и вывода изображений.
java. awt. image	В данном пакете содержатся средства для манипулирования с изображениями, получаемыми по сети.
Java.applet	Данный класс имеет множество полезных методов, поскольку является основой для всех апплетов и может также при помощи интерфейса Appietcontext предоставлять информацию об окружении апплета.

Java Enterprise API

Java Enterprise API обеспечивает взаимодействие с корпоративными базами данных. При помощи данного API-интерфейса корпоративные разработчики могут строить распределенные клиент-серверные апплеты и приложения на Java, работающие в любых ОС или аппаратных платформах, имеющихся в компании.

Java Commerce API

Интерфейс Java Commerce API обеспечивает создание защищенных коммерческих и финансовых приложений в сети Web. JavaWallet является компонентом начального уровня, он описывает и реализует клиентскую платформу для программ, работающих с кредитными и дебетными картами и электронными платежами.

Java Server API

Интерфейс Java Server API — это масштабируемая платформа, позволяющая легко разрабатывать разнообразные Java-совместимые серверы Internet и intranet.

Java Media API

Модуль Media Framework имеет часы для синхронизации и медиаплейеры для воспроизведения аудио-, видео- и MIDI-файлов. Модули 2D и 3D обеспечивают развитые средства обработки изображений. Для создания движущихся и трансформирующихся 2D-объектов можно применять анимацию. Модуль Java Share обеспечивает совместное использование приложений многими пользователями; пример такого приложения — коллективная "белая доска". И наконец, модуль Telephony позволяет интегрировать телефон и компьютер.

Java Security API

Java Security API — шифрование с цифровыми подписями, кодирование и проверка прав на доступ.

Java Management API

Интерфейс Java Management API располагает большим набором масштабируемых Java-объектов и методов для построение апплетов, могущих управлять корпоративными сетями через Internet и intranet-сети.

Java Beans API

Java Beans API — это набор API-интерфейсов для создания программных компонентов. К примеру, компонент "кнопка" может запустить создание диаграммы в другом компоненте. Модули Java Beans можно подключать к компонентам Microsoft OLE/COM/Active-X, OpenDoc и Netscape LiveConnect.

Java Embedded API

Является подмножеством стандартного Java API для встроенных устройств, полностью поддерживающих Java Core API. Данный интерфейс включает минимальный встраиваемый API, построенный на базе классов java.lang, java.util и, частично, java.io. Кроме того, имеются некоторые расширения для определенных задач, например, для работы в сети и графических интерфейсов.

8. Основные конструкции языка Java

8.1. Файлы классов, описание класса

Программы в Java строятся на объектах. Объекты создаются на основании классов. Класс – это описание свойств и методов объекта: каждый класс – в отдельном файле. Классы могут дополнительно объединятся в пакеты, ориентируясь по назначению классов. Синтаксис описания класса приведен ниже:

Синтаксис: модификатор_доступа спецификаторы **class** имя_класса **extends** имя_класса_родителя **implements** имя_интерфейса { тело класса: его свойства и методы }
Примеры: 1) public final **class** MyPlan **extends** Plan **implements** Administration { /* свойства класса, методы класса */ } 2) class Prosto { // свойства и методы класса }

Здесь скобки "{" и "}", называемые "начало блока" и "конец блока", означают начало и конец класса. Блоки { } используются в программах на Java для обозначения начала и конца любой конструкции или участка кода (например, конструкции IF..ELSE, цикла FOR и т.д.). Каждый оператор программы на Java внутри скобок { }, т.е. внутри описания класса или метода должна заканчиваться на точку с запятой ";".
Конструкция /* */ означает начало и конец комментариев. Комментарии могут добавляться и при помощи символов // , которые означают, что, начиная с этого места и до конца строки, идет комментарий.
Необходимо также помнить, что в Java различаются маленькие и большие буквы, так что MyPlan – это не то же самое, что и myPlan.

Значение модификаторов доступа класса приведено ниже:

Значение	Описание
не указано	Класс доступен только для объектов, находящихся в том же пакете.
public	Класс доступен для всех объектов. Класс public должен обязательно содержаться в файле, имеющем такое же название, что и имя класса.

Спецификаторы класса могут быть следующими:

Значение	Описание
final	Запрещено создание подклассов, на основании этого класса. Используется по соображениям безопасности, чтобы, например, нельзя было подменить менеджер безопасности, создав на его основе собственный объект и переопределив методы объекта.
abstract	Абстрактный класс (хотя бы один метод объявлен, но не описан).

* Спецификаторы final и abstract несовместимы, т.е. класс не может быть одновременно final и abstract.

Описание класса – это просто описание свойств и методов объекта. Для того, чтобы объектом можно было пользоваться, необходимо создать экземпляр класса, т.е. создать объект, на основании имеющегося описания. Об этом будет сказано ниже, а сейчас рассмотрим, как описываются свойства и методы класса.

8.2. Типы данных, свойства класса, модификаторы доступа свойств и методов, массивы.

В Java используются следующие типы данных:

Тип	Описание
boolean	Имеет значения true или false (ИСТИНА или ЛОЖЬ).
byte	8-бит целое число со знаком в диапазоне от -128 до 127
short	16-бит целое число со знаком в диапазоне от –32 768 до 32 767
char	16-бит Unicode-символы или цифровые значения от 0 до 65535.
int	32-бит целое число со знаком в диапазоне от –2 147 483 648 до 2 147 483 647
long	64-бит целое число в диапазоне от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, причем такая точность сохраняется на любом компьютере, независимо от платформы.
float	32-бит число с плавающей точкой обычной точности ($\approx\pm10^{39}$)
double	64-бит число с плавающей точкой двойной точности ($\approx\pm10^{317}$)
String	Строковая константа. После объявления строки ее содержимое нельзя изменять.
StringBuffer	Строковый буфер. Позволяет изменять содержимое строки.

* Внимание! Регистр символов ВАЖЕН. Неверно: "Int", "string". Верно: "int", "String".

Описание свойства (переменной) происходит следующим образом:

Синтаксис: модификатор_доступа спецификаторы тип_данных имя_свойства = первоначальное_значение ;
Примеры: 1) public **int** primer **= 100;** // Создается public свойство с именем primer, типа int, и значением 100. 2) private static **byte** primer2, primer3, primer4; // Описаны три свойства, типа byte.

Модификаторы доступа свойства совпадают с модификаторами доступа методов и означают следующее (версия языка Java2):

Модификатор доступа	В рамках одного пакета		Из других пакетов	
	наследование	доступ	наследование	доступ
не указан	да	да	нет	нет
public	да	да	да	да
protected	да	да	да	нет
private	нет	нет	нет	нет

Здесь "наследование" означает возможность порождать от этого класса другие классы, наследуя его свойства и методы (в порожденных классах их не надо заново описывать), а "доступ" означает возможность создавать экземпляр класса, на основании этого класса.

Одновременно с модификаторами доступа, возможно использование спецификаторов свойств:

Значение	Описание
final	Задает константу (значение переменной не меняется в процессе выполнения программы). Первоначальные значения всех свойств final необходимо определить при их объявлении: final int Primer = 5;
static	Модификатор static делает свойство (переменную) класса статическим. Статическое свойство – это свойство, принадлежащее не экземплярам класса, а самому классу. При создании экземпляров класса, каждый экземпляр получает свою "независимую копию" всех не статических свойств и методов. В разных экземплярах класса, не статические свойства и методы могут иметь разные значения. Статические свойства и методы принадлежат не экземплярам класса, а самому классу, поэтому, при создании экземпляров класса, экземпляры не получают "копии" статических свойств и методов, а, следовательно, значения статических свойств будут одинаковы для всех экземпляров класса. Доступ к статическим свойствам происходит либо через имя экземпляров класса (изменение статического свойства в любом экземпляре класса приведет к его синхронному изменению для остальных экземпляров класса), либо через имя самого класса (можно получить доступ к статическим свойствам, не создавая ни одного экземпляра класса). Поясним сказанное на примере. Пусть существует класс "Лифт", в котором описаны два свойства: "напряжение" и "этаж", причем свойство "напряжение" имеет модификатор static. Пусть на основании класса "Лифт", создано два экземпляра класса (два объекта): "Левый_лифт" и "Правый_лифт". Теперь, если перерубить силовой кабель, то оба лифта будут одновременно обесточены, но остановятся на разных этажах: Лифт.напряжение = 0, Левый_лифт.напряжение = 0, Правый_лифт.напряжение = 0 – общее статическое свойство для всех лифтов, однако свойства Левый_лифт.этаж и Правый_лифт.этаж будут различаться.

* Переменная может быть одновременно static и final.

Свойство может быть описано, как на уровне класса (глобальная переменная), так и на уровне метода. Свойство уровня класса описывается вне тела методов и существует, пока существует экземпляр класса. Все методы класса могут обращаться к такому свойству. Свойство уровня метода описывается внутри самого метода и существует только пока выполняется метод. Обращаться к нему из других методов нельзя. Ниже приведен пример, в котором свойство bigBrother описано на уровне класса, а свойство smallBrother описано на уровне метода, с именем Metod. Пример:

```
Class Primer {
        long  bigBrother;                 // Свойство описано на уровне класса
        void Metod (  )  {
              short smallBrother;         /* Свойство описано на уровне метода */ }  }
```

Массивы:

Индексация в массиве начинается с 0. Индекс имеет тип "int". Таким образом, команда Massiv [3] [3] = 4; присваивает значение элементу массива, находящемся в 4-ой строке и 4-ом столбце.

Объявление массива с заданием первоначальных значений:

Синтаксис: модификатор_доступа тип_данных имя_массива [ ] = {значение1, значение2, значениеN};

Пример: int Massiv [ ] = {123,17,29,15 }; // одномерный массив

public long Massiv [ ] [ ] = { {123,17,29,15 }, {17, 12, 3} };   // двумерный массив

Объявление массива, с последующим его созданием:

Синтаксис: модификатор_доступа тип_данных имя_массива [ ];
имя_массива = new тип_данных [число_элементов];

Пример: private int Massiv [ ];   // одномерный массив
Massiv = new int [4];

long Massiv [ ] [ ];   // двумерный массив
Massiv = new long [4] [3];

Объявление массива, одновременно с его созданием:

Синтаксис: модификатор_доступа тип_данных имя_массива [ ] = new тип_данных [число_элементов];

Пример: int Massiv [ ] = new int [4];   // одномерный массив
private protected long Massiv [ ] [ ] = new long [4] [3];   // двумерный массив

8.3. Методы класса, методы доступа в классах, конструкторы и деструкторы класса.

Объявление метода

Синтаксис: модификатор_доступа спецификаторы тип_возвращаемого_значения имя__метода (параметры) **throws** список_исключений_которые_вызывает_класс { /* операторы */ return возращаемое_значение ; }
Пример: 1) public static **long** Metod (int x, byte y) throws IOException { /*операторы*/   return 5; } 2) void Metod ( ) { /*операторы*/   return; /*операторы*/}

* Во втором примере тип_возвращаемого_значения = void. Тип void – это "пустой" тип, что означает, что метод не возвращает значений. При указании типа void, ключевое слово return указывается без параметров (или вообще не используется) и просто означает преждевременный выход из метода.

Модификаторы доступа методов (нет, public, protected, private) – см. модификаторы доступа свойств.

Спецификаторы методов:

Значение	Описание
static	Задает статический метод. Подробнее объяснения см. в спецификаторе static для свойств.
abstract	Абстрактные методы просто объявляются, но не реализуются в данном классе. Тело метода должно быть описано в подклассах текущего класса. Ни static-методы, ни конструкторы классов не могут объявляться как abstract. Более того, абстрактные методы нельзя определять как final, поскольку в этом случае их нельзя будет переопределить.
final	Любые подклассы текущего класса не смогут переопределить данный метод (о переопределении методов см. ниже). Эта возможность увеличивает защищенность ваших классов и гарантирует, что операции, определенные в данном методе, никак нельзя будет изменить.
native	Методы, написанные не на языке Java, а на других языках программирования. Обычно такие методы пишутся на C++ для ускорения работы критических участков программы. Для определения этих методов нужно поместить спецификатор native в начале объявления метода и вместо тела метода поставить точку с запятой. Пример: native void toggleStatus();
synchronized	Позволяет защитить данные, которые могут быть разрушены в том случае, если два метода, из разных потоков, пытаются одновременно обратиться к одним и тем же данным. Syncronized-метод не может начать работать со свойствами, пока их не освободит другой метод.

Методы доступа в классах.

<u>*Свойства*</u> класса можно создавать, описывая на уровне класса, public - переменные. Однако, более оптимальное решение – это описать на уровне модуля private-переменную, а для чтения или записи ее значения использовать public-методы, называемые методами доступа. Метод доступа позволяет, перед записью нового значения свойства, выполнить определенные действия: проверить значение свойства на корректность, автоматически пересчитать значения связанных свойств и т.д. Пример:

```
class Krug {
private int radius;
        public void setRadius (int new_radius) {
        if (new_radius>0) {radius=new_radius;} else { System.out.println ("Некорректное значение радиуса"); }
        }
        public int getRadius ( ) { return ( radius ); }
```

}

Конструкторы класса, перегрузка методов.
Конструкторы класса – это специальные методы, которые вызываются при создании экземпляров классов. Обычно в этих методах содержатся операторы, которые позволяют проинициализировать значения свойств объекта при его создании. Для инициализации собственных свойств объекта используется ключевое слово **this** (этот объект). Синтаксис: this.свойство = значение;

Можно определять несколько конструкторов с различными наборами параметров. Тогда, при создании экземпляра класса, будет возможность создавать объект, передавая ему различные наборы параметров (перегружая конструкторы). При этом будет вызываться тот конструктор, у которого именно такой набор параметров, который был указан при создании объекта. Перегружать можно не только конструкторы, но и любые методы класса.

Имя конструктора должно совпадать с именем класса. Для конструктора не указывается тип возвращаемых им значений, т.к. конструктор не может возвращать каких либо значений. Модификатор доступа конструктора всегда должен быть public. Конструкторы нельзя объявлять, как native, absract, static, synchronized или final. Пример:

```
class Game {
boolean vse_svoi; // Описание свойства "все свои"
public Game (String Name) {this.vse_svoi=true; } /* инициализация свойства при создании объекта */
public Game (int Chislo_igrokov) { this.vse_svoi=false; } /* инициализация свойства при создании объекта */
public Game (String Name, int Chislo_igrokov) {/* операторы, выполняются при создании объекта */ }
}
```

Деструкторы класса (метод finalize)
Метод finalize присутствует во всех объектах Java (он порожден от объекта java.lang.Object, от которого порождены все объекты Java). Этот метод выполняется непосредственно перед "сборкой мусора", т.е. перед тем, как объект будет уничтожен. Пример:

```
class Game {     protected void finalize ( ) { System.out.println ("Игра окончена"); }      }
```

8.4. Создание экземпляра класса

Описание класса – это просто описание свойств и методов объекта. При создании же экземпляра объекта (ключевое слово new), объект создается реально: анализируется описание класса, выделяется необходимая память под объект, объект получает собственные копии всех не static свойств и методов класса и, в зависимости от переданных создаваемому классу параметров, срабатывает тот или иной конструктор класса, который выполняет заданные программистом операции по инициализации свойств объекта. Можно создавать несколько экземпляров одного класса. При создании экземпляра класса, в используемые переменные помещается указатель на созданный объект (см. переменные domino и shashki в примере ниже). На один и тот же объект может существовать несколько указателей (см. переменную domino2 в примере ниже). Объект существует до тех пор, пока на него есть хотя бы одна ссылка, иначе он автоматически уничтожается Java ("сборка мусора").
Синтаксис создания экземпляра класса следующий:
Вариант 1 (предварительное описание, с последующим созданием):
имя_описанного_класса имя_переменной;
имя_переменной = new имя_описанного_класса (параметры_для_конструктора);
Вариант 2 (создание в момент описания):
имя_описанного_класса имя_переменной = new имя_описанного_класса (параметры_для_конструктора);
Пример (предполагается, что класс Game описан раньше (см. выше)):

```
public Class dvor {
        Game domino, domino2;                    // Описание переменных на уровне класса
public dvor ( ) {                                //Конструктор класса dvor
        domino = new Game ("Домино", 6);         /* Создается игра "домино" на 6 человек, срабатывает
                                                    соответствующий конструктор класса */
        Game shashki = new Game ("Шашки");  /* Создается игра "шашки", число участников и так известно.
                                                    Т.к. список параметров другой, срабатывает конструктор, у
                                                    которого перечень параметров именно такой */
        domino2 = domino;   /* За спинами доминошников появляются "болельщики", которые также лезут в
                               игру – переменная domino2 указывает на тот же объект, что и domino.*/
        shashki = domino;   /* В шашки играть бросают, и присоединяются к "болельщикам" доминошников:
                               на объект "домино" уже 3 указателя: domino, domino2 и shashki, а объект
                               "шашки" будет уничтожен при ближайшей "сборке мусора", т.к. на него нет
                               ни одного указателя */
}
```

```
public static void main(String args[ ] ) {                // Выполнение программы всегда начинается с метода main
dvor go = new dvor ( ); // Вызов конструктора dvor, который и выполнит основную часть программы
System.gc ( ); // Принудительная "сборка мусора"  ( gc = Garbage Collection )
}}
```

Приведенная выше программа начнет выполняться с метода main (как и любая исполняемая программа на Java), который вызовет создание экземпляра класса dvor. При создании экземпляра класса, будет вызван конструктор dvor ( ), который и выполнит основную часть программы. Такой прием применен из-за того, что метод main – статический, а из статического метода нельзя вызывать не статические методы и свойства других объектов (например, создавать объекты domino и shashki).

8.5. Наследование, переопределение методов

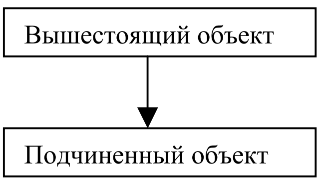
Мощной возможностью Java, как и любого объектно-ориентированного языка, является наследование. При наследовании, порожденный класс получает свойства и методы того класса, от которого он был порожден, и их не надо переписывать заново. Кроме того, он может дополнительно определить свои собственные свойства и методы, а также переопределить унаследованные методы (т.е. реализовать те же методы, но по своему). Наследование задается при помощи ключевого слова **extends** в описании класса (синтаксис см. раньше). Переопределение унаследованных методов реализуется путем описания в порожденном классе метода с таким же именем, что и унаследованный метод. После того, как метод переопределен, доступ к первоначальному варианту возможен с использованием специальной переменной **super** (доступ: super.метод_родителя или super.свойство_родителя), которая указывает на класс, от которого был порожден данный. Ниже приведен пример, в котором от класса avtomobil порождается класс mercedes, в котором переопределен метод poehal и создан новый метод marka. Причем метод poehal переопределен так, что сначала вызывается старый вариант метода (super.poehal ( )), а затем добавляются новые команды. Кроме того, предусмотрена возможность перегрузки метода, т.е. вызова одного и того же метода, с различными вариантами параметров. Пример:

```
class avtomobil {
        long probeg;       // Задаем свойство "пробег"
        void poehal ( ) { System.out.println (" Еду "); }  // Задаем метод "поехал" }

class mercedes extends avtomobil  {          // Объект mercedes порожден от avtomobil
    /* Свойство probeg определять не надо, т.к. оно унаследовано. Метод poehal также не надо было бы
    определять, но мы хотим заменить реализацию этого метод на свою, т.е. переопределить метод.
    Кроме того, мы создаем возможность перегружать метод poehal, описывая различные реализации этого
    метода для различных наборов параметров */
        void poehal ( )  {      super.poehal( );        System.out.println (" Еду быстро …");   }
        void poehal ( String kuda )  {      super.poehal( );        System.out.println (" Еду "+kuda);  }
        void marka ( )  { System.out.println (" Шестисотый мерс!"); }  //Задаем новый метод marka
        }
public class proga {
        public proga( ) {
        mercedes myCar = new mercedes ( );
        myCar.marka ( ); myCar.poehal( );  myCar.poehal( " в клуб"); }

public static void main(String args[ ] ) { proga go = new proga (); }
}
```

8.6. Создание иерархии объектов.



Необходимо отличать наследование, от создания иерархии объектов. При наследовании, порожденный класс получает набор свойств и методов от класса, от которого происходит наследование. Таким образом, эти два класса имеют много общего. При создании иерархии объектов, вышестоящий и подчиненный объекты могут не иметь ничего общего, но один объект подчиняется другому, т.е. доступ к свойствам и методам подчиненного объекта возможен через свойство вышестоящего объекта. Проводя аналогию с жизнью: сын похож на отца, но начальник и подчиненный не обязательно из одной семьи. Хочется также подчеркнуть уже прозвучавшую разницу: наследование – для классов, иерархия – для объектов, т.е. для экземпляров класса. Иерархия задается путем описания в классе вышестоящего объекта свойства, создающего подчиненный объект, в классе которого может быть также описано свойство, создающее объект еще более низкого уровня иерархии и т.д. Тогда, создав вышестоящий объект, вы автоматически создаете дерево подчиненных объектов. В примере ниже, описан объект mercedes, который содержит подчиненный объект koleso (свойство osi),

который, в свою очередь, содержит подчиненный объект болт (свойство zaklepki). При создании автомобиля автоматически создаются колеса и болты. Доступ к подчиненным объектам происходит в формате: вышестоящий_объект . подчиненный_объект . свойства_подчиненного_объекта.

```
package primer;
class avtomobil {
long probeg = 3;
void poehal ( ) { System.out.println (" Еду "); }
}
class mercedes extends avtomobil  {                          // Наследование от класса avtomobil
void poehal ( )  { System.out.println (" Еду быстро "); }
void marka ( )  { System.out.println (" Шестисотый мерс!"); }
koleso osi = new koleso ();                                  // Создание подчиненного объекта
}
class koleso {
boolean zapaska=false;          // Описание обычного свойства "запаска"
bolt zaklepki = new bolt();       // Создание подчиненного объекта
}
class bolt { int nomer_bolta = 100; }

public class proga {
mercedes myCar; // Описание переменной myCar

public proga() {
myCar = new mercedes ( );  // Создание вышестоящего объекта
myCar.marka ( );              // Использование метода вышестоящего объекта
myCar.osi.zapaska = true;   // Использование свойства объекта второго уровня иерархии
myCar.osi.zaklepki.nomer_bolta = 1;  // Использование свойства объекта третьего уровня иерархии
}

public static void main(String args[ ] ) {  // Запуск программы
proga go = new proga ();}
}
```

8.7. Использование пакетов

Для того, чтобы в одном классе можно было использовать другой класс, необходимо импортировать этот класс при помощи директивы **import**. Пример: **import** avtomobil; . Однако вы видите, что во всех приведенных ранее примерах этого не сделано. Объясняется это тем, что директиву import можно опустить, если файл класса находится в том же каталоге. Директива import используется, в основном, для импорта классов из пакетов.

Рассмотрим понятие пакета. Как известно, каждый класс помещается в отдельный файл класса. Большое количество файлов классов напоминает жесткий диск без каталогов или папок, поэтому файлы классов должны быть некоторым образом организованы. Для объединения исходных текстов файлов классов в один файл используются пакеты. Они напоминают библиотеки, имеющиеся во многих языках. Помимо того, что пакеты позволяют логически сгруппировать классы, они еще снимают проблему совпадения имен классов. Допустим, что есть два пакета "маляры" и "художники" и в рамках каждого пакета описан класс "кисточка" (очевидно, что малярная кисть отличается от кисти художника). Если бы пакетов не существовало, то возник бы конфликт, т.к. существует два класса с одинаковым именем. Использование пакетов снимает эту проблему, т.к. теперь доступ к классам будет организован так: "маляры.кисточка" и "художники.кисточка".

Для того чтобы включить класс в пакет, нужно при его объявлении использовать оператор **package**, который должен быть первым оператором в файле. Другими словами, перед этим оператором могут быть комментарии и пробелы, но ничего больше. Пример:

Верно	Неверно
package tachki; **import** java.awt.*; **import** java.io.*; class avtomobil { long nomer; void poehal ( ) { System.out.println (" Еду "); } } class mercedes **extends** avtomobil { void poehal ( ) { System.out.println (" Еду быстро …"); } void marka ( ) { System.out.println (" Шестисотый мерс!"); } }	**import** java.awt.*; **import** java.io.*; **package** tachki; class avtomobil { long nomer; void poehal ( ) { System.out.println (" Еду "); } } class mercedes **extends** avtomobil { void poehal ( ) { System.out.println (" Еду быстро …"); } void marka ( ) { System.out.println (" Шестисотый мерс!"); } }

Для использования класса из пакета, необходимо импортировать его, или импортировать пакет целиком.
Синтаксис:
import имя_пакета . имя_класса // импортировать один класс
import имя_пакета . * // импортировать все классы из пакета
Пример:
import tachki . mercedes;
import tachki . * ;
Можно и не импортировать пакет, тогда для использования класса придется указывать его полное название, включая имя пакета, что неудобно: tachki.mercedes moya_tachka = new tachki.mercedes ( );
Также желательно не импортировать большие пакеты целиком, а ограничиться только теми классами, которые действительно нужны:
1) При импорте пакета целиком виртуальная машина Java должна отслеживать имена всех элементов, имеющихся в пакете. На это расходуется дополнительная оперативная память. Кроме того, в этом случае немного снижается быстродействие системы
2) Если импортируется несколько пакетов и окажется, что они имеют совпадающие имена файлов классов, то возникнет конфликт: какой же класс все-таки использовать?
3) Самый главный минус связан с пропускной способностью Internet-соединения. Когда вы целиком импортируете пакеты, отсутствующие на вашем компьютере, то перед тем, как продолжить работу, браузер должен загрузить через сеть все файлы классов для целого пакета. Если в пакете имеется 30 классов, а используются только два, будут напрасно расходоваться значительные ресурсы и время пользователя.

8.8. Интерфейсы

В языке Java отсутствует множественное наследование, т.е. любой объект может порождаться только от одного объекта. Но иногда надо, чтобы объект унаследовал не только свойства и методы родителя, но и свойства и методы другого объекта. Для этих целей используются интерфейсы – классы, в которых объявлены, но не описаны методы. Класс, реализующий интерфейс, должен сам описать реализацию методов интерфейса. При описании методов интерфейсов, можно не указывать модификаторы доступа, все они по умолчанию рассматриваются как public. Другие модификаторы доступа в описании методов интерфейса использовать нельзя. В интерфейсах также нельзя использовать переменные, не являющиеся константами (final). Любые поля интерфейса, вне зависимости указаны модификаторы доступа или нет, рассматриваются как public, final и static.

Синтаксис объявления интерфейса следующий: public **interface** имя_интерфейса **extends** список_интерфейсов { /* объявление свойств, объявление методов без описания их реализации */ }
Пример: public interface Radio { final String nazvanie = "Автомобильный радиоприемник"; // описание константы public long Volna( ); // описание метода }

Интерфейс применяется к объекту при помощи ключевого слова implements (синтаксис см. выше). Пример: class mercedes **extends** avtomobil **implements** Radio – к классу mercedes применяется интерфейс Radio.

8.9. Арифмитические и логические операции. Циклы и управляющие операторы.

Арифметические и логические операции

Операция	Описание
x = 3	Присвоить переменной x значение 3.
+, -, *, /, %	Сложение, вычитание, умножение, деление и деление по модулю (определение остатка от деления).
x ++, x --	Увеличить x на единицу, уменьшить x на единицу.
+=, -=, *=, /=	Сложение, вычитание, умножение и деление с присваиванием. x+=3 аналогично команде x=x+3.
x == 3	Операция равенства: x равен 3. Используется в конструкции IF..ELSE и др. Пример: if (x==3) { //действия };
x != 3	Операция неравенства: x не равен 3.
<, <=, >, >=	Операции отношений: меньше, меньше или равно, больше, больше или равно.
x | y	Поразрядная операция ИЛИ (OR). Результатом является число, полученное в соответствии с таблицей истинности для ИЛИ. Пример: x: 00001010 = 10 y: 00001100 = 12 Результат: 00001110 = 14
x & y	Поразрядная операция И (AND).
x ^ y	Поразрядная операция исключающее ИЛИ (XOR).
~ x	Поразрядное логическое отрицание НЕ (NOT).
&=, |=, ^=	Поразрядные операции с присваиванием: x&=y аналогично x=x&y
x || y	Логическая операция ИЛИ (OR). В отличие от поразрядных операций, результат логической операции не число, а только значение ИСТИНА или ЛОЖЬ. Именно логические операции используются в конструкциях IF..ELSE и др.
x && y	Логическая операция И (AND).
!x	Логическая операция НЕ (NOT).
-x	Унарная операция "изменение знака".
x<<2	Поразрядный сдвиг битов влево: сдвинуть биты переменной x на 2 бита влево. Пример: 00000001 (двоичное) <<2 дает 00000100 (двоичное).
>>, >>>	Поразрядный сдвиг вправо, поразрядный сдвиг вправо без знака.
<<=, >>=, >>>=	Поразрядный сдвиг с присваиванием.

Оператор if - else:

Синтаксис: if   (выражение) {операторы;}   else   {операторы; }
Пример: 1) if (x==2 && x!=3) { /*операторы выполняются если x равен 2 и не равен 3*/ } else { /*операторы */ } 2) if (x==2) { /*операторы */ }

Оператор while

Синтаксис: 1) while (выражение) {операторы;} 2) do {операторы;} while (выражение);
Пример: 1) while(x==2 && x!=3) { x++; /* и др. операторы - выполняются пока x равен 2 и не равен 3 */ } 2) do { /*операторы */ } while (x==2);

Операторы for

Синтаксис: 1) for (тип_данных счетчик=исходное_значение; (условие_остановки); шаг) { /*операторы */ }
Пример: 1) for (int i=0, long j=10;   ((i >100) || yslovie_J ( ));   i++, raschet_J ( )) { /*операторы */ }

2) for (int i=0; i==100; i++) {/*операторы */ }

Описание меток
В Java нет операторов Goto, поэтому управление на метки передается при помощи операторов **break** и **continue** (см. ниже). Метки описываются не относительно всей программы, а относительно только тех операторов, которые заключены между скобками { } в описании метки. Описание меток можно вкладывать одно в другое.

Синтаксис: метка :   { /*операторы, относительно которых определена метка*/ }
Пример: metka1: { x++;   metka2:{ y++; }   }

Оператор break
Используется в конструкциях while, for и switch (см. ниже). Прерывает выполнение цикла и передает управление по указанной метке. Если метка не указана, то управление передается строке, следующей за циклом. Пример:

Пример 1	metka1: { for (int i=0; i==100; i++) { metka2: { for (int m=0; m==100; m++) { if (i==77) break metka1; if (m==99) break metka2; } }}}
Пример 2	for (int i=0; i==100; i++) { if (i==77) break; }
Пример 3	metka: { int x = 3; /* операции */ if (x==3) break metka; }

Операторы continue
Используется в конструкциях while, for и switch (см. ниже). Прерывает выполнение цикла и преждевременно передает управление к следующему проходу цикла. Указание метки в операторе continue позволяет выбрать уровень вложенности цикла, которому передается управление. Пример:

Пример 1	metka1: { for (int i=0; i==100; i++) { metka2: { for (int m=0; m==100; m++) { if (i==77) continue metka1; if (m==99) continue metka2; } }}}
Пример 2	for (int i=0; i==100; i++) { if (i==77) continue; }

Операторы switch
Переключатель switch (должен иметь целый тип: byte, short, char, int, но не float и boolean) передает управление одному из нескольких составляющих его операторов в зависимости от значения переключателя. Если совпадений нет, то управление передается метке по умолчанию (default). Если default-метка отсутствует, программа продолжается с первого оператора, стоящего после блока switch.

Синтаксис: switch (переключатель) {   case значение1 : оператор1; операторN; break;   case значениеN: оператор1; операторN; break;   default: оператор1; операторN; break; }
Пример: int x=1; switch (x){ case 1: System.out.println ("one"); break; case 2: case 3: case ' ': System.out.println ("two or three or space"); break; case default: System.out.println ("Default"); }

8.10. Математические функции, дата и время, работа со строками

Математические функции:
В пакете java.lang имеется класс Math, который предоставляет некоторые полезные математические и тригонометрические функции. Пример:

```
class primer {
public static void main(String args[ ] ) {  // Запуск программы
```

```
int chislo1 = Math.min (100, 700); // Возвращает минимальное значения из двух чисел 100 и 700
int chislo2 = Math.max (100, 700); // Возвращает максимальное значения из двух чисел 100 и 700
int chislo3 = Math.abs (-3); // Возвращает абсолютное значение числа
int chislo4 = ( int ) (   10*Math.random ( )    ) + 1;  /* Возвращает случайное целое число от 1 до 10.
     Здесь Math.random( ) возвращает случайное число от 0 до 1, а конструкция ( int ) ( выражение ) служит
     для преобразования выражения, находящегося под скобками, к типу int. Аналогично можно проводить
     преобразования и между другими типами данных. Наример: ( long ) ( 15.234 + 10.017 ) */
long chislo5 =  Math.round (10.5);  //Округление по правилам математики. Результат = 11
double chislo6 =  Math.floor(10.7);  //Округление всегда в меньшую сторону. Результат = 10
double chislo7 =  Math.ceil (10.3);  //Округление всегда в большую сторону. Результат = 11
double chislo8 =  Math.rint (10.5);  //Округление до ближайшего целого. Результат = 10
double chislo9 =  Math.sqrt ( 4 ); //Квадратный корень из 4.
double chislo10 =  Math.pow ( 7, 1/3 ); // Число 7 в степени 1/3
double chislo11 =  Math.log ( 7 ); // Натуральный логарифм числа 7.
double chislo12 =  Math.log ( 7 ) / Math.log ( 10 ); // Десятичный логарифм числа 7.
double chislo14 =  Math.exp ( 7 ); // Экспонента числа 7, т.е. e^7.  Существует константа Math.E
double chislo15 =  Math.sin ( 3.14 ); // Синус. Угол задается в радианах. Радианы=градусы*Math.PI / 180.
double chislo16 =  Math.cos ( Math.PI / 2 ); // Косинус. Угол задается в радианах.
double chislo17 =  Math.tan ( 3.14 ); /* Тангенс. Существуют также методы asin(), acos(), atan(), означающие
                                         арккосинус, арксинус и арктангенс, соответственно. */
} }
```

Для генерации случайных чисел можно воспользоваться и классом Random из пакета java.util. Пример:

```
import java.util.Random
class primer {
public static void main(String args[ ] ) {  // Запуск программы
Random generator1 = new Random (12345 );  /* Инициализирует генератор случайных чисел числом 12345.
                                         Если повторно проинициализировать генератор этим числом, то будет
                                         получена такая же последовательность случайных чисел. Если не
                                         задать число, то для инициализации выберется значение системного
                                         таймера (в миллисекундах, прошедших от 01.01.1970 г.)*/
int chislo1 = generator1.nextInt ( );  // 32-разрядное случайное число типа int
long chislo2 = generator1.nextLong ( ); // 64-разрядное случайное число типа long
float chislo3 = generator1.nextFloat ( ); // случайное число типа float в диапазоне от 0 до 1
double chislo4 = generator1.nextDouble ( ); // случайное число типа double в диапазоне от 0 до 1
double chislo5 = generator1.nextGaussian ( );  /* Случайное число нормального распределения. Среднее
                                                  значение =0, стандартное отклонение = 1 */
}}
```

Дата и время:
Для работы со значениями даты и времени можно воспользоваться следующими конструкциями:

```
import java.util.Date  // почти все функции работы с датой и временем находятся в этом классе
class primer {
public static void main(String args[ ] ) {  // Запуск программы
long timer1 = System.currentTimeMillis( ); //Число миллисекунд, прошедших с 01.01.1970
Date timer2 = new Date ();                  // Создание объекта "дата"
int  vremya1 = timer2.getYear();            // Получить число лет, прошедших с 1900 г, т.е. для 2001 = 101
int  vremya2 = timer2.getMonth();           // Получить текущий месяц. Январь = 0, Февраль = 1 и т.д.
int  vremya3 = timer2.getDate();            // Получить текущую дату
int  vremya4 = timer2.getHours();           // Получить часы
int  vremya5 = timer2.getMinutes();         // Получить минуты
int  vremya6 = timer2.getSeconds();         // Получить секунды
int  vremya7 = timer2.getDay();             // Получить день недели: 0-воскресенье, 1-понедельник и т.д.

// Если менеджер безопасности разрешит вам эти операции, то вы сможете модифицировать дату и время:
timer2.setYear(102); // Установить 2002 год
timer2.setMonth(11); // Установить 12-й месяц
timer2.setDate(20);
timer2.setHours(23);
timer2.setMinutes(5);
timer2.setSeconds(10);
```

}}

Работа со строками:

Работа со строками в Java реализована не достаточно удобно. Строки в Java фактически представляют собой строковые константы: после объявления строки ее содержимое нельзя изменять. Для изменения самой строки нужно создать объект StringBuffer, которому можно присвоить значение строки, а потом его модифицировать. Затем можно создать новую строку, равную значению объекта StringBuffer. Пример:

Программа	Комментарии
String stroka="Это";	// Или более строго: String stroka =new String("Это");
StringBuffer karman =new StringBuffer (stroka);	// Создание буфера с именем karman
karman.insert(0, 'a') ;	// Вставка "а" в 1 позицию строки (индексация - с нуля)
karman.append ('!') ;	// Добавление "!" к строке
karman.setCharAt (2, 'a');	// Замена третьего символа на "а"
stroka=new String (karman);	// Присваивание нового значения строке.

Единственное исключение из такой неудобной процедуры - можно складывать строки, например:

```
String komy = "вам" ;
String summa = "Привет "+komy+"братья" ;
```

Над строками можно выполнять некоторые функции, в результате которых создаются новые измененные строки (примеры см. ниже). Переносить часть строки на новую строчку нельзя. В тексте строки нельзя использовать двойные или одинарные кавычки и косую черту "\". Если же необходимо этими символами все-таки воспользоваться, то применяют управляющие текстовые и восьмеричные константы:

Константа	Значение
\n	перевод строки
\f	перевод формата
\r	возврат каретки
\"	\u0022 двойная кавычка
\'	\u0027 одиночная кавычка

Константа	Значение
\\	\u005c обратная косая черта
\007	\u0007 звонок
\101	\u0041 буква"A"
\071	\u0039 цифра "9"

Пример:

```
String primer = "На разных \n строках и \" в кавычках \" буква \110, со звонком \007"
```

Строки – это объекты, поэтому сравнивать строки при помощи простой операции равенства нельзя, т.к. при этом сравниваются не содержимое строк, а их объекты (которые будут разными, даже если содержимое строк одинаково). Для сравнения строк надо пользоваться специальными методами.

Пример:

```
String stroka1 = “Это строка”;                    // создание объекта stroka1
String stroka2 = “Это строка”;                    // создание объекта stroka2
if (stroka1==stroka2) {/* операторы */}  // условие не выполнится, т.к. stroka1и stroka2 – это 2 разных объекта
if ( stroka1.equals(stroka2) ) {/*операторы*/ }   // а вот так сравнивать строки правильно (метод equals)
boolean otvet = stroka.equalsIgnoreCase (stroka2); // сравнение без учета регистра.
```

Для поиска слова в словаре можно воспользоваться методом compareTo ( ), который возвращает:
 0 - строки равны, число >0 – строка больше сравниваемой, число <0 – строка меньше сравниваемой.

Пример:

```
String slovo = “Сергей”;                   // создание объекта slovo
int otvet1 = slovo.compareTo('Сам');       // результат: otvet1>0
int otvet2 = slovo.compareTo('Сеялка'); /* результат: otvet2<0  - значит слово "Сергей" находится по алфавиту
                                             между словами "Сам " и "Сеялка". */
```

Над строками можно выполнять и другие операции. Ниже приведены примеры некоторых функций для работы со строками:

```
String stroka = “Это строка”;  // создание строки stroka
boolean otvet = stroka.startsWith(‘Это’); /* переменная otvet будет true (истина), если строка начинается со
                                              слова “Это”, регистр учитывается */
boolean otvet = stroka.endsWith(‘строка); /* переменная ответ будет true (истина), если строка кончается
                                              словом “строка”, регистр учитывается */
int  otvet = stroka.indexOf('o');  // переменная otvet будет содержать номер позиции первой буквы "о" в строке
int  otvet = stroka.indexOf('o',otvet+1);  /* после того, как в предыдущем примере будет найдена первая буква
                                            "о", в этом примере поиск следующей буквы "о" в строке будет
                                            продолжен, начиная со следующей после "о" позиции. Так можно
                                            найти все буквы "о", продолжая поиск до тех пор, пока otvet != 0*/
```

```
int  otvet = stroka.lastIndexOf('o');  // тоже самое, что и indexOf('o'), но поиск начинается с конца строки
int  otvet = stroka.lastIndexOf('o', otvet-1 );  // аналогично примеру выше
int  otvet = stroka.indexOf('Это');  /* можно определять позицию не только отдельного символа, но и
                                        подстроки */
char bukva = stroka.charAt (3); /* в переменную bukva будет помещен 4-й символ строки (индекс первого
                                   символа строки = 0) */
char massiv [ ] = new char [20];  // создается массив из 20-ти символов
stroka.getChars(5,10,massiv, 2); /* строка с 6 по 11 символ, посимвольно помещается в массив, начиная с 3-ей
                                    позиции в массиве. Существует аналогичный метод getBytes ( ). */
String stroka2 = new String ( massiv [ ], 3, 2); /* создается новая строка длиной 2 символа, значения для строки
                                                   берутся, начиная с 4-ой позиции в массиве */
String stroka2 = stroka.subString(4); /* выделение подстроки из строки "stroka", начиная с 5-ой позиции и до
                                         конца строки */
String stroka2 = stroka.subString(4 ); /* выделение подстроки из строки "stroka", начиная с 5-ой позиции и до
                                          конца строки */
String stroka2 = stroka.subString(4, 9 ); /* выделение подстроки из строки "stroka", начиная с 5-ой и
                                             заканчивая 10-ой позицией строки */
String stroka2 = stroka.replace('o', 'a'); // заменяет в строке все буквы "o" на букву "a"
String stroka2 = stroka.toUpperCase( ); // преобразует строку в верхний регистр
String stroka2 = stroka.toLowerCase( ); // преобразует строку в нижний регистр
int chislo=10;  String stroka = String.valueOf(chislo); // переводит практически любой тип данных в тип String
```

Использование класса String Tokenizer (разбивка строк на слова):

Иногда необходимо разбить введенную пользователем строку на отдельные слова (например, при анализе запроса к поисковой системе). Для этого в пакете java.util определен специальный класс StringTokenizer, который разбивает строук на слова, ориентируясь на пробелы в качестве разделителей. Пример программы с использованием этого класса приведен ниже:

```
class primer {
import java.util.StringTokenizer;
public static void main(String args[ ] ) {                    // Запуск программы
String stroka = "Инструкция по пользованию мылом";     // Задаем строку для разборки
StringTokenizer  razborka = new StringTokenizer (stroka);  // Создаем объект "razborka", типа StringTokenizer
System.out.println ("Число слов в строке = "+ razborka.countTokens( ) );   /* метод countTokens ( ) подсчиты-
                                                                             вает число слов в строке */
System.out.println ("Строка содержит следующие слова: " );
while (razborka.hasMoreTokens ( ) )              // метод hasMoreTokens возвращает true, пока в строке есть слова
System.out.println (razborka.nextToken ( ) );   // метод nextToken возвращает следующее слово в строке
}
```

8.11. Блоки try .. catch .. finally, обработка исключений.

При возникновении ошибки, в Java программе возбуждается исключение (exception). Ошибки можно обрабатывать при помощи конструкции try .. catch .. finally.

Синтаксис: **try** { /* часть программы, где возможно возникновение ошибки */ } **catch** (тип_исключения переменная) { /* блок кода, выполняющийся при возникновении ошибки; */ } **finally** { /* блок кода, выполняющийся после блока try и блока catch, даже если ошибки не было */}
Пример 1: **try** { /* часть программы, где возможно возникновение ошибки */ } **catch** (NumberFormatException oshibka) { String text="Неверный формат числа :"+oshibka.toString; System.out.println(text); } **catch** (ArithmeticException oshibka) { String text="Ошибка вычислений:"+oshibka.toString; System.out.println(text); } **finally** { System.out.println("Конец"); }
Пример 2*: **import** java.io. IOException **try** { /* часть программы, где возможно возникновение ошибки */ } **catch** (IOException oshibka) { System.out.println("Какая-то ошибка ввода-вывода"); }

* Перехват различных типов ошибок определен в различных пакетах (классах), поэтому перед заданием блока **try .. catch .. finally** необходимо импортировать **import** соответствующий класс из соответствую-

щего пакета (см. пример 2). Однако часть исключений описана в пакете java.lang - их можно использовать сразу, т.к. этот пакет уже "проимпортирован" для любой Java программы.

Некоторые типы исключений:

Исключение	Пакет	Причина возникновения
RuntimeException	java.lang	Ошибка времени выполнения.
SecurityException	java.lang	Апплет пытается выполнить действие, запрещенное режимом защиты, установленным в браузере
OutOf MemoryException	java.lang	Недостаток памяти при размещении нового объекта
StackOverflowException	java.lang	Переполнение стека
ArithmeticException	java.lang	Математические ошибки, например, деление на ноль
NumberFormatException	java.lang	Неправильное преобразование между строками и числами. Неверный формат числа
IndexOutOfBoundsException	java.lang	Индекс вышел за пределы структуры
ArrayIndexOutOfBoundsException	java.lang	Неправильные индексы массива
StringIndexOutOfBoundsException	java.lang	Программа пыталась обратиться к несуществующей позиции символа в строке
ArrayStoreException	java.lang	Программа пытается записать в массив неправильный тип данных
NullPointerException	java.lang	Ссылка на null-объект
NoSuchMethodException	java.lang	Объект не имеет такого метода.
IOException	java.io	Общие ошибки ввода/вывода, например, невозможность чтения из файла
FileNotFoundException	java.io	Обращение к несуществующему файлу
EOFException	java.io	Достигнут конец файла
InterruptedIOException	java.io	Ввод/вывод прерван
AWTException	java.awt	Ошибка оконного графического интерфейса
MalformedURLException	java.net	Неверно задан URL-адрес
ProtocolException	java.net	Ошибка протокола.
SocketException	java.net	Ошибка сокета
UnknownHostException	java.net	Хост не найден
UnknownServiceException	java.net	Сервис не найден

Переменной oshibka, в приведенном выше примере, присваивается указатель на объект "исключение", который имеет следующие методы:
getMessage () – возвращает строку, с подробной информацией об исключении.
toString() – преобразует объект в строку, которую можно вывести на экран.
printStackTrace () – отображает иерархию вызовов методов, приведших к исключению.

9. Создание приложений на языке Java, запуск приложений

Исполняемая программа должна состоять, по крайней мере, из одного класса, содержащего метод main, с которого и начинается выполнение программы. Метод main должен быть описан следующим образом: public static void main(String args[ ]) { /* Операторы */ }. Пример простейшей программы приведен ниже:

```
package  games;
import   java.awt.*;  //  импорт пакета просто ради примера

class Game {
public Game (String Name) {
        String messaga="Начинается игра"+ Name;
        System.out.println (messaga); }
protected void finalize ( ) { System.out.println ("Игра окончена"); }
}

public class dvor {
Game domino;
public dvor ( ) {
domino = new Game ("Домино");
}
public static void main ( String args[ ] ) {
dvor  zapusk = new dvor ( ); }
```

}
Не откомпилированная программа на Java хранится в файле с расширением java. После компиляции программой javac.exe из пакета JDC или любым другим компилятором получается большое количество файлов с расширением class, которые содержат байт-код соответствующих классов программы. Для запуска программы достаточно набрать java.exe ИмяКласса, где java – это название исполняемого файла, содержащего виртуальную Java-машину, а ИмяКласса – это имя public класса, содержащего метод main. Естественно, в текущем каталоге должен находится файл " ИмяКласса.class", а также другие файлы ".class" программы. Для удобства, все файлы ".class" программы часто объединяют в zip, cab или jar архивы: это не архивы как таковые, а просто способ собрать в единое целое программу, разбитую на отдельные файлы. В случае jar – архива запуск программы будет выглядеть как java.exe –jar ИмяАрхива.jar. В среде Windows можно упростить запуск программы, создав соответствующий ярлык. Можно также установить, что для запуска файлов, с расширением "jar" используется приложение "java.exe –jar", тогда можно будет просто запустить программу, щелкнув по ней два раза в проводнике.

10. Создания апплетов на языке Java

Любой класс, описывающий апплет, должен быть порожден от класса java.Applet. Пример:
import java. Applet. *;
public **class** My_Applet **extends** Applet {/* свойства и методы апплета */}
Апплет выполняется не самостоятельно, а в контексте браузера. Поэтому выполнение апплета начинается не с метода main (апплет вообще не должен содержать метод main), а с методов, описывающих цикл жизни апплета. Апплет имеет жизненный цикл, состоящий из пяти этапов:

1) **Этап инициализации (init).** Система создает и загружает объект апплета. Инициализация выполняется только один раз. Для того, чтобы выполнить свои действия на этом этапе, необходимо переопределить метод init ( ), унаследованный от класса java.Applet (пример см. ниже).
2) **Этап запуска (start).** Система начинает выполнение апплета. В отличие от этапа инициализации, этап запуска на протяжении жизненного цикла может выполняться множество раз. Обычно в браузере, для ускорения работы, кэшируются предыдущие просмотренные страницы (например, 4 последние). Так что, уходя со страницы, пользователь не выгружает ее из памяти, а просто помещает в кэш. Нажатие кнопки "назад" просто активизирует, имеющуюся в памяти страницу, и ее не надо будет снова загружать по сети. При каждой такой повторной активизации, метод Init не выполняется, а метод Start выполняется. Для выполнения своих команд на этом этапе, необходимо переопределить метод start ( ), унаследованный от класса java.Applet (пример см. ниже).
3) **Этап прорисовки (paint).** Этот этап выполняется каждый раз, когда область отображения апплета должна быть прорисована на экране, т. е. сразу же после этапа запуска апплета, а также всякий раз, когда изображение апплета восстанавливается или изменяется. Это происходит, когда окно апплета освобождается от перекрывающего его другого окна, когда пользователь, прокручивая страницу, возвращается к изображению апплета, или когда программа явно перерисовывает окно апплета (метод repaint ( )). Для выполнения своих команд на этом этапе, необходимо переопределить метод paint (Graphics g) (пример см. ниже). Метод paint не описан в классе Applet. Класс Applet наследует этот метод от класса Component, который является суперклассом в длинной цепочке наследования, следующей от класса Applet к классам Panel, Container и Component. Для переопределения метода paint необходимо импортировать библиотеки java.awt.*, содержащие информацию о классе Graphics.
4) **Этап останова (stop).** Система выполняет фазу останова, когда пользователь обращается к другой Web-странице, деактивизируя текущую страницу (но реально не выгружая ее, а оставляя в памяти). Этап остановки может выполняться множество раз. По умолчанию, на этом этапе апплет продолжает работу в фоновом режиме. Для выполнения своих команд на этом этапе, необходимо переопределить метод stop ( ), унаследованный от класса java.Applet (пример см. ниже).

выполняется только один раз и соответствует выгрузке web-страницы из памяти. Если апплет использовал ресурсы, которые перед уничтожением апплета необходимо освободить, то это нужно делать на этапе уничтожения. Для выполнения своих команд на этом этапе, необходимо переопределить метод destroy ( ), унаследованный от класса java.Applet (пример см. ниже).

Пример переопределения методов, описывающих жизненный цикл апплета:

```
import  java. Applet. *;  // необходимо для любого апплета
import java. awt. * ;     // необходимо для переопределения метода paint
public class MyApplet extends Applet {
        public void init ( )  { /* Здесь располагается ваш код этапа инициализации */ }
        public void start ( ) { /* Здесь располагается ваш код этапа запуска */ }
        public void paint ( Graphics g ) { /* Здесь располагается ваш код этапа перерисовки */ }
        public void stop( ) { /* Здесь располагается ваш код этапа останова */ }
        public void destroy( ) {/* Здесь располагается ваш код этапа уничтожения */ }
```

}
Обратите внимание на то, что для переопределения метода paint необходимо импортировать библиотеки java.awt.*, содержащие информацию о классе Graphics. Описанная в примере переменная g , указывающая на объект класса Graphics, позволяет выводить информацию и графику в области отображения апплета (или на так называемом холсте). Пример:

```
Import java.net.URL;  // Необходимо для загрузки рисунка

public void paint ( Graphics g ) {
// Пример отображения текста
Font shrift = new Font ("TimesRoman", Font.BOLD+Font.ITALIC, 14); /* жирный курсив, 14-й шрифт Times.
                                        Возможные варианты - шрифт: TimesRoman, Courier, Symbol,
                                        Dialog, Helvetica, аттрибуты: PLAIN, BOLD, ITALIC  */
g.setFont (shrift);                      // установка шрифта для вывода в окне апплета
g.drawString ("Строка",  100, 50);       // вывод текста в позиции x=100 y=50

// Пример отображения рисунка (поддерживаются только GIF и JPEG форматы)
URL address= getDocumentBase (); //если рисунок находится в каталоге документа HTML
URL address= getCodeBase (); //если рисунок находится в каталоге файла class апплета
Image risunok = getImage (adress, "podkatalog / my_risunok.gif");  // загрузка рисунка
g.drawImage (risunok, 100, 500, this); // отображение рисунка в позиции X:Y = 100:500

// Пример установки цвета
g.setColor(Color.white);                 /* Установка белого цвета. Возможны варианты:
            white, lightGray, gray, darkGray, black, red, pink, orange, yellow, green, magneta, cyan, blue */
g.setColor(Color.green.darker( ) );     g.setColor(Color.green.brighter( ) );  //цвета темнее или ярче
// существует также метод getColor ( ), позволяющий получить текущий цвет

// Пример установки цвета в формате RGB
Color cvet = new Color (255, 192, 192);
g.setColor (cvet);

// Пример установки цвета фона
setBackground(Color.red);

// Пример отображения линии, прямоугольника и круга (элипса)
g.drawLine ( 100, 200, 400, 500 ); /* рисует линию от точки с координатами X:Y = 100:200, до точки с
                                    координатами 400:500 */
g.drawRect (150, 100, 200, 120); /* не закрашенный прямоугольник: 150:100 – X:Y верхнего левого угла,
                                    200 – ширина, 120 – высота */
g.fillRect (150, 100, 200, 120);  // закрашенный прямоугольник
g.clearRect (150, 100, 200, 120);  // очищает прямоугольную область
g.draw3DRect (150, 100, 200, 120); // не закрашенный объемный  прямоугольник
g.fill3DRect (150, 100, 200, 120);  // закрашенный объемный прямоугольник
g.drawOval (150, 100, 200, 120) /* не закрашенный элипс: 150:100 – X:Y верхнего левого угла, 200 –
                                 ширина, 120 – высота прямоугольника, в который будет вписан элипс */
g.fillOval (150, 100, 200, 120);  // закрашенный элипс. Если ширина равна высоте – получится круг.
g.drawRoundRect (150, 100, 200, 120, 10,15); /* прямоугольник с закругленными краями. 10:15 - отступы
                                         от угла прямоугольника, с которых начинается скругление */
g.fillRoundRect (150, 100, 200, 120, 10,15); // закрашенный прямоугольник с закругленными краями.
}
```

В примере приведены только простейшие возможности использования объекта Graphics. Для более детальной информации обращайтесь к книгам по Java.

10.1. Менеджер расположения

Апплет может содержать такие элементы управления, как кнопки, флажки, раскрывающиеся списки и т.д. Подробнее об этом будет сказано ниже. Однако, прежде чем переходить к описанию синтаксиса этих элементов, необходимо рассмотреть понятие "менеджер расположения".

Для различных компьютеров трудно заранее предсказать точные координаты того места, где должен находится элемент управления. Действительно, если бы положение элементов управления жестко задавалось в абсолютных координатах, то при запуске в Windows 95 с дисплеем 640x480 апплета, рассчитанного на X-терминал с разрешением 1280x1024, получилось бы неизвестно что. Поэтому размещением элементов управления в апплете занимается менеджер расположения, которому программист только указывает, как те

или иные элементы должны быть расположены по отношению к другим элементам. Точные значения координат вычисляет сам менеджер. Существуют пять различных типов менеджеров расположения:

1) **Flow Layout** (последовательное расположение). Располагает элементы последовательно слева направо, пока они помещаются в одном ряду. Затем переходит на следующий ряд, и так далее. Этот менеджер установлен в апплете по умолчанию.
2) **Grid Layout** (табличное расположение). Представляет контейнер как таблицу, состоящую из клеток одинакового размера. Располагает элементы в клетках таблицы, начиная с левого верхнего угла и продолжая вправо и вниз, подобно последовательному расположению. Различие между последовательным и табличном расположением состоит в том, что при табличном расположении четко задается число элементов в строке (или столбце), а всем элементам отводятся клетки одинаковой формы и размера.
3) **Border Layout** (полярное расположение). Рассматривает апплет как лимб компаса. При добавлении нового элемента менеджеру расположения сообщается, что элемент следует поместить в одну из пяти областей: "север" (North), "юг" (South), "запад" (West), "восток" (East) и "центр" (Center). В каждой области может быть не более одного элемента. Точные координаты вычисляются, исходя из относительных размеров элементов.
4) **Card Layout** (блокнотное расположение). Все элементы, входящие в контейнер, рассматриваются как странички блокнота. В каждый данный момент видна только одна из страничек.
5) **Grid Bag Layout** (ячеистое расположение). Наиболее универсальный менеджер расположения и в то же время наиболее запутанный. Менеджер Grid Bag Layout разделяет контейнер на клетки равного размера, как и GridLayout. Отличие GridBagLayout состоит в том, что он сам определяет, сколько требуется строк и столбцов, а также позволяет элементу занимать при необходимости более одной клетки. Область, занимаемая элементом, называется его ячейкой (display area). Прежде чем добавить элемент в апплет, надо задать менеджеру GridBagLayout набор "пожеланий" насчет того, где следует размещать элемент. Эти пожелания выдаются в форме объекта GridBagConstraints, который имеет следующие свойства:
   - **gridx и gridy.** Координаты клетки, куда будет помещен следующий элемент (если он будет занимать более одной клетки, это координаты левой верхней занимаемой клетки). Левый верхний угол в объекте GridBagLayout имеет координаты (0, 0). По умолчанию переменные gridx и gridy имеют значение GridBagConstraints.RELATIVE. Для gridx это означает клетку непосредственно справа от последнего добавленного элемента, а для gridy — клетку, примыкающую к последнему элементу снизу.
   - **gridwidth и gridheight.** Число клеток, которое занимает компонент по горизонтали и вертикали. По умолчанию оба эти значения равны 1. Если требуется, чтобы компонент был последним в строке, следует задать gridwidth равным GridBagConstraint.REMAINDER (тоже самое для gridheight означало бы, что компонент является последним в столбце). Если компонент должен стоять предпоследним в строке или столбце, следует задать GridBagConstraint.RELATIVE.
   - **fill.** Сообщает как поступить, если элемент меньше, чем предоставленная ему ячейка. По умолчанию используется значение GridBagConstraint.NONE, которое оставляет размер элемента без изменений. GridBagConstraint .HORIZONTAL заставляет растянуть элемент по горизонтали до размеров ячейки, без изменения вертикального размера. GridBagConstraint.VERTICAL заставляет растянуть элемент по вертикали до размеров ячейки без изменения горизонтального размера. GridBagConstraint.BOTH указывает, что элемент следует растянуть в обоих направлениях до размеров ячейки.
   - **ipadx и ipady.** Указывает, сколько пикселов добавить к размерам компонента по осям x и y. Добавление происходит с каждой стороны элемента, так что параметр ipadx, равный 4, приведет к увеличению размера элемента на четыре пиксела вправо и на четыре влево. По умолчанию используется значение 0.
   - **insets**. Экземпляр класса Insets. Указывает, сколько места нужно оставить между границами элемента и краями ячейки. Другими словами, параметр insets создает "демаркационную линию", окружающую элемент. В примере, приведенном ниже, создается экземпляр класса Insets с отступами в 20 пикселей со всех сторон: Insets myInsets = new Insets (20,20,20,20);
   - **anchor**. Используется тогда, когда размеры элемента меньше размеров его ячейки. Этот параметр указывает, где должен располагаться элемент в пределах ячейки. По умолчанию применяется GridBagConstraint.CENTER (разместить в центре), но есть еще несколько значений: GridbagConstraints.NORTH (север), GridBagConstraints.NORTHEAST (северо-восток), GridBagConstraints.EAST (восток), GridBagConstraints.SOUTHEAST (юго-восток), GridBagConstraints.SOUTH (юг), GridBagConstraints.SOUTHWEST (юго-запад), GridBagConstraints.WEST (запад), GridBagConstraints.NORTHWEST (северо-запад).
   - **weightx и weighty.** Применяются для задания относительных размеров элементов. Например, элемент с параметром weightx, равным 2.0, занимает по горизонтали вдвое большее пространство, чем элемент с weightx, равным 1.0. Поскольку это относительные значения, неважно, имеют ли

все элементы значение параметров 1.0 или 3.0. Следует задать эти параметры хотя бы для одного компонента по каждому из направлений, так как иначе менеджер GridBagLayout сожмет все компоненты к центру апплета.

Примеры задания менеджеров расположения:

```
FlowLayout myManager = new FlowLayout( ); // создание менеджера типа FlowLayout
FlowLayout myManager = new FlowLayout(FlowLayout.RIGHT,10,5); /* создание менеджера FlowLayout,
                                                выравнивание по правому краю, расстояние между элементами
                                                по горизонтали – 10 пикселей, вертикали – 5 пикселей */
setLayout (myManager); // активизация менеджера myManager
```

```
GridLayout myManager = new GridLayout (0, 4); // 4  колонки для элементов, число строк - автоматически
GridLayout myManager = new GridLayout (4, 0); // 4  строки для элементов, число колонок – автоматически
GridLayout myManager = new GridLayout (4, 2); /* 4  строки для элементов, число колонок игнорируется и
                                                определяется автоматически */
GridLayout myManager = new GridLayout (4, 0, 10, 5); /* расстояние между элементами по горизонтали – 10
                                                пикселей, вертикали – 5 пикселей */
setLayout (myManager); // активизация менеджера myManager
```

```
BorderLayout myManager = new BorderLayout ( ); // создание менеджера типа BorderLayout
BorderLayout myManager = new BorderLayout (10,5); /* установка менеджера BorderLayout, промежуток
                                между элементами по горизонтали – 10, по вертикали – 5 пикселей. */
Button myButton = new Button ("Нажми меня"); // создание кнопки myButton
setLayout (myManager); // активизация менеджера myManager
this.add(myButton, myManager.CENTER); // добавление кнопки myBtton в апплет (this), в область "Центр"
```

```
GridBagLayout myManager = new GridBagLayout( ); // создание менеджера типа GridBagLayout
Button myButton = new Button("Нажми меня"); // создание кнопки myButton
setLayout(myManager); // активизация менеджера myManager
GridBagConstraints ogranichenie = new GridBagConstraints( ); // создание объекта ограничений
ogranichenie.weightx = 1.0; // описание ограничений
ogranichenie.gridwidth = GridBagConstraints.RELATIVE; // описание ограничений
ogranichenie.fill = GridBagConstraints.BOTH; // описание ограничений
myManager.setConstraints(myButton, ogranichenie ); // активизация ограничений для кнопки myButton
this.add (myButton); // добавление кнопки myButton в апплет
```

Все приведенные выше менеджеры расположения – это классы, описанные в пакете AWT. Существуют также и другие менеджеры расположения. Например, в пакете com.borland.jbcl.layout, поставляемом вместе со средой разработки Jbilder, описан менеджер расположения XYLayout, который позволяет самостоятельно задавать размеры и координаты элементов:

```
import com.borland.jbcl.layout.*;              // импорт пакета, содержащего менеджер расположения
XYLayout myManager = new XYLayout( ); // создание нового менеджера расположения
this.setLayout(myManager);                     // активизация менеджера для апплета (this)
myManager.setWidth(283); myManager.setHeight(261);          // ширина и высота окна менеджера
Button myButton = new Button("Knopka");                    // создание кнопки с надписью Knopka
this.add ( myButton, new XYConstraints (128, 191, 123, 25) ); // добавление кнопки в координаты X:Y=128:191
                                                           // и с размером кнопки 123 на 25 пикселей
```

10.2. Элементы управления в апплете, обработка событий.

Апплет может содержать различные элементы управления: кнопки (класс Button), флажки и переключатели (класс Chekbox), списки (класс List), выпадающие списки (класс Choice), надписи (класс Label), поля ввода (класс TextField), текстовые области (класс TextArea), раскрывающиеся и контекстные меню и др. элементы. Большинство из них определено в пакете AWT, однако возможно использование и других пакетов, например, javax.swing, com.borland.dbswing и др.

Каждый элемент управления, и апплет в целом, имеет свой список событий, происходящих при тех или иных действиях пользователя. Достаточно просто описать методы, которые будут выполняться при наступлении тех или иных событий. Примеры событий: mousePressed (кнопка мыши нажата), mouseReleased (кнопка мыши отпущена), mouseClicked (щелчок кнопкой мыши), mouseEntered (мышь вошла в границы элемента), mouseExited (мышь вышла за пределы элемента), mouseMoved (мышь перемещается над элементом), mouseDragged (мышь перемещается над элементом, удерживая нажатой кнопку мыши), keyPressed (клавишу нажали – можно определить какая), keyReleased (клавишу отпустили), keyTyped (обобщенное событие: клавиша нажата и отпущена), focusGained (элементом получен фокус),

focusLost(элементом потерян фокус), actionPerformed (обобщенное событие: произошло действие над элементом), itemStateChanged (изменилось состояние элемента – для флажков) и др.

Добавление элементов управления в апплет происходит, обычно, на стадии инициализации апплета (метод Init) по следующей общей схеме:

1) Создать и активизировать менеджер расположения

```
XYLayout myManager = new XYLayout( );
this.setLayout(myManager);    myManager.setWidth(283);    myManager.setHeight(261);
```

2) Создать элемент управления

```
Button button1 = new Button( );
```

3) Добавить элемент управления в апплет

```
this.add(button1, new XYConstraints(128, 191, 123, 25));
```

4) Описать процедуры обработки событий для элемента.

```
//установить "прослушиватель" событий мыши для кнопки button1
button1.addMouseListener(new java.awt.event.MouseAdapter() {
        // при нажатии кнопки мыши вызываем метод button1_mousePressed
        public void mousePressed(MouseEvent e) {button1_mousePressed(e);}
        // при отпускании кнопки мыши вызываем метод button1_ mouseReleased
        public void mouseReleased(MouseEvent e){button1_mouseReleased(e);}
        // при щелчке кнопкой мыши вызываем метод button1_ mouseClicked
        public void mouseClicked(MouseEvent e) {button1_mouseClicked(e);}
        // конец "прослушивателя" событий мыши для кнопки button1
  });
```

5) При возникновении событий, пользоваться свойствами и методами элемента, для манипулирования его состоянием и получения введенной пользователем информации.

```
// вызов этих процедур был описан на шаге 4
void button1_mousePressed(MouseEvent e) {
button1.setBackground(Color.red);  // при нажатии на кнопку она становится красной
}
void button1_mouseReleased(MouseEvent e) {
button1.setBackground(Color.gray); // при отпускании кнопки она становится опять серой
}
void button1_mouseClicked(MouseEvent e) {
button1.setLabel("knopka was clicked by user"); // после щелчка на кнопке, надпись на ней изменяется
}
```

** Внимание: при описании переключателей, учитывайте следующие особенностью пакета AWT: фактически в AWT есть только флажки (квадратик с галочкой), но отсутствуют переключатели (кружок с точкой). Для того, чтобы флажок стал переключателем, необходимо создать группу переключателей (класс CheckboxGroup), а затем добавить флажок к этой группе (в любой момент, только один из переключателей в группе может быть включен). Пример: CheckboxGroup Group1 = new CheckboxGroup(); Checkbox checkbox1 = new Checkbox(); checkbox1.setCheckboxGroup(Group1);*

Большинство сред разработки программ на Java позволяет добавлять эти элементы визуально, а система сама будет писать исходный код, который потом необходимо будет только чуть-чуть подправить. Ниже приведен пример апплета, созданного таким образом:



Исходный текст апплета приведен ниже:

```
import java.applet.*;                 // пакет необходим для любых апплетов
import java.awt.*;                    // пакет содержит элементы управления
import java.awt.event.*;              // необходимо для обработки событий
import com.borland.jbcl.layout.*;  // пакет содержит менеджер расположения XYLayout

public class easy extends Applet {
  boolean isStandalone = false;      // апплет не может выполняться самостоятельно (в отдельном окне)
  XYLayout myManager = new XYLayout();           // создание менеджера расположения
  List list1 = new List();                                   // создание списка
  Button button1 = new Button("knopka");             // создание кнопки
  TextArea textArea1 = new TextArea();               // создание области для ввода текста
  Label label1 = new Label();                              // создание надписи
  TextField textField1 = new TextField();              // создание текстового поля
  CheckboxGroup Group1 = new CheckboxGroup(); // создание группы переключателей
  Checkbox checkbox1 = new Checkbox();             // создание переключателя 1
  Checkbox checkbox2 = new Checkbox();             // создание переключателя 2
  Checkbox checkbox3 = new Checkbox();             // создание переключателя 3
  Checkbox checkbox4 = new Checkbox();             // создание флажка 4 (не будет входить в группу Group1)
  TextField info = new TextField();         // текстовое поле (можно Label ) для вывода служебной информации

public easy() {  /* пустой конструктор апплета */}

public void init() { // инициализация апплета, вызывается метод jbInit и отслеживаются ошибки
try {jbInit();} catch(Exception e) {e.printStackTrace();}    }

private void jbInit() throws Exception {      // метод, инициализирующий апплет
    this.setLayout(myManager);                  // установка менеджера расположения
    myManager.setWidth(283); myManager.setHeight(261); // // ширина и высота окна менеджера

    // добавление элементов управления в апплет (this)
    this.add(textArea1, new XYConstraints(144, 13, 121, 105));
    this.add(textField1, new XYConstraints(105, 129, 161, 24));
    this.add(checkbox1, new XYConstraints(17, 167, 76, 15));
    this.add(checkbox2, new XYConstraints(101, 168, 76, 15));
    this.add(checkbox3, new XYConstraints(185, 168, 76, 15));
    this.add(checkbox4, new XYConstraints(19, 193, 85, 22));
    this.add(label1, new XYConstraints(17, 125, 84, 32));  this.add(button1, new XYConstraints(128, 191, 123, 25));
    this.add(info, new XYConstraints(4, 228, 275, 26));      this.add(list1, new XYConstraints(8, 12, 123, 108));

    this.setBackground(Color.pink);                              // установка цвета фона апплета
    this.setFont(new java.awt.Font("DialogInput", 1, 14));       // установка шрифта апплета

    // добавление "прослушивателя" движений мышью над апплетом
    this.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
      public void mouseMoved(MouseEvent e) {this_mouseMoved(e);}
      public void mouseDragged(MouseEvent e) {this_mouseDragged(e);}
    });
    //  добавление "прослушивателя" щелчков мышью по апплету
    this.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseClicked(MouseEvent e) {this_mouseClicked(e);}
    });

button1.setBackground(Color.orange); button1.setForeground(Color.blue); // оранжевый фон, синий текст кнопки
button1.setFont(new java.awt.Font("SansSerif", 3, 14));  // установление шрифта кнопки
   // добавление "прослушивателя" щелчков мышью по кнопке
  button1.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mousePressed(MouseEvent e) {button1_mousePressed(e);}
      public void mouseReleased(MouseEvent e){button1_mouseReleased(e);}
      public void mouseClicked(MouseEvent e) {button1_mouseClicked(e);}
    });
   // добавление "прослушивателя" движений мышью над апплетом
    button1.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
```

```
    public void mouseMoved(MouseEvent e) {button1_mouseMoved(e);}  });

   list1.setBackground(Color.white);   list1.setForeground(Color.blue);  // белый фон, синие буквы в списке
   list1.addItem("Иванов"); list1.addItem("Петров"); list1.addItem("Сидоров");  // добавление в список фамилий
   // добавление "прослушивателя" событий списка, срабатывает при двойном щелчке по элементу списка
   list1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {list1_actionPerformed(e);} });

   textArea1.setBackground(Color.white);    textArea1.setForeground(Color.blue);  // белый фон, синие буквы
   textArea1.setText("Текстовое поле"); // текст, отображаемый в поле ввода

   label1.setFont(new java.awt.Font("Dialog", 1, 14));  label1.setText("Надпись"); // форматирование надписи

   // описание поля ввода
   textField1.setBackground(Color.white); textField1.setForeground(Color.blue);
   textField1.setComponentOrientation(null);  textField1.setText(" поле ввода");
   // добавление "прослушивателя" событий поля ввода
   textField1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {textField1_actionPerformed(e);}  });

   // изменение подписей переключателей и добавление их в группу Group1
   checkbox1.setLabel("флажок1");  checkbox1.setCheckboxGroup(Group1);
   checkbox2.setLabel("флажок2");  checkbox2.setCheckboxGroup(Group1);
   checkbox3.setLabel("флажок3");  checkbox3.setCheckboxGroup(Group1);
   checkbox2.setEnabled(false);     // сделать переключатель 2 недоступным для пользователя
   Group1.setCurrent(checkbox1); // по умолчанию, в группе выбран первый переключатель

   checkbox4.setLabel("флажок4");  // надпись для флажка 4
   checkbox4.setState(true);              // установить флажок 4 в значение истина (есть галочка)
   // добавление "прослушивателя" событий изменения состояния флажка 4
   checkbox4.addItemListener(new java.awt.event.ItemListener() {
   public void itemStateChanged(ItemEvent e) {checkbox4_itemStateChanged(e);} });
}

public void start() {  }       // пустые стадии жизненного цикла апплета
public void stop() {  }       // пустые стадии жизненного цикла апплета
 public void destroy() {  } // пустые стадии жизненного цикла апплета
```

// Вызов обработчиков событий, приведенных ниже, задан в "прослушивателях" событий

```
void this_mouseClicked(MouseEvent e) {  info.setText("Щелкнули по апплету");}
void this_mouseMoved(MouseEvent e) {  info.setText("");  }
void this_mouseDragged(MouseEvent e) { info.setText("Что-то тянете :)");    }

void button1_mousePressed(MouseEvent e) { info.setText("Вы нажали кнопку");  }
void button1_mouseReleased(MouseEvent e) { info.setText("Вы отпустили кнопку");  }
void button1_mouseClicked(MouseEvent e) { /* ничего не делать */ }
void button1_mouseMoved(MouseEvent e) {  info.setText("Вы над кнопкой"); }

void checkbox4_itemStateChanged(ItemEvent e) {
  if (checkbox4.getState()) {info.setText("Флажок 4 установлен");}
  if (!checkbox4.getState()) {info.setText("Флажок 4 снят");}
}

void list1_actionPerformed(ActionEvent e) {
  info.setText(list1.getSelectedItem());  // выводит содержимое выбранного пользователем элемента списка
}

void textField1_actionPerformed(ActionEvent e) {
  info.setText(textField1.getText());      // выводит введенный пользователем текст
}}
```

10.4. Фреймы, меню, диалоговые окна.

Обычно апплет выполняется непосредственно в контексте HTML-страницы. Однако апплет может выполняться и в отдельном окне, называемом фрейм. Фрейм создается на основании класса Frame из пакета AWT. Последовательность действий при создании фрейма такова:

1) Создать пустой фрейм.
2) Задать для фрейма менеджер расположения и размеры окна фрейма.
3) Создать апплет, который будет отображаться в фрейме.
4) Проинициализировать и запустить апплет.
5) Добавить апплет во фрейм.
6) Отобразить фрейм (метод show).

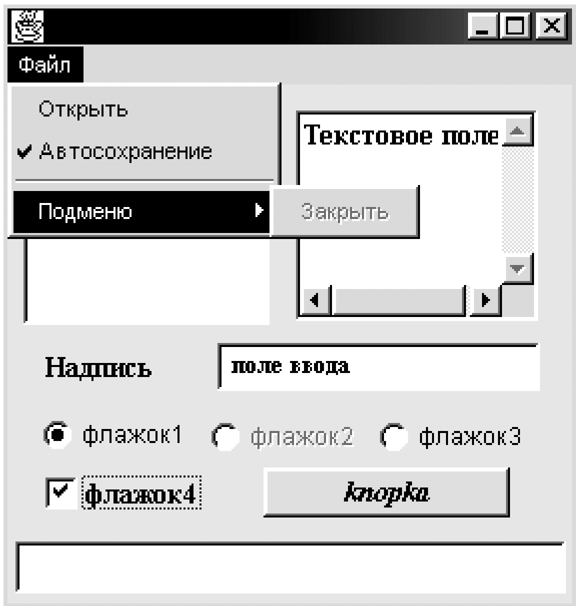
Фрейм может иметь собственное меню (классы: MenuBar – строка меню, Menu – меню, MenuItem – команда меню, CheckboxMenuItem – команда меню, которую можно отмечать флажком). Для создания меню необходимо:

1) Создать строку меню и добавить ее в фрейм.
2) Создать меню и добавить его в строку меню.
3) Создать команду меню, описать для нее обработчик события и добавить ее в меню.
4) Создать вложенные подменю и добавить их в меню.

Фрейм может создавать и использовать собственные диалоговые окна. Диалоговое окно создается только на основании фрейма. Апплет не может непосредственно создать диалоговое окно. Диалоговое окно может быть модальным – пока оно не будет закрыто, работа с фреймом будет невозможна. Диалоговое окно создается на основании класса Dialog. Для создания диалогового окна необходимо:

1) Создать диалоговое окно.
2) Задать для диалогового окна менеджер расположения.
3) Добавить в окно элементы (кнопки, надписи, флажки и т.д.) и описать обработчики событий элементов.
4) Подогнать размер диалогового окна под отображаемые элементы (метод pack) и запретить изменение размеров окна.
5) Отобразить диалоговое окно (метод show).

Апплет, выполняющийся в фрейме, может запускаться и без участия браузера – для этого в апплете необходимо описать метод main. Ниже приведен пример апплета, выполняющегося в фрейме, содержащего строку меню и отображающего диалоговое окно перед началом работы.




```
import java.awt.*;
import java.awt.event.*;
import com.borland.jbcl.layout.*;

public class Frame1 extends Frame {

public Frame1( ) {/*пустой конструктор фрейма*/  }
```

```
public static void main(String[] args) {
 final Frame1 myFrame = new Frame1( );           // создание фрейма
 final easy myApplet = new easy();               // создание апплета для отображения в фрейме
   myApplet.init(); myApplet.start();            // инициализация и запуск апплета
   myFrame.setLayout(new XYLayout( ) );          // задание менеджера расположения для фрейма
   myFrame.add(myApplet);                        // добавление апплета в фрейм
   myFrame.resize(290,290);                      // изменение размера фрейма

   MenuBar myBar = new MenuBar( );               // создание строки меню
   myFrame.setMenuBar(myBar);                    // добавление строки меню в фрейм
   Menu fileMenu = new Menu ("Файл");            // создание меню "Файл"
   myBar.add(fileMenu);                          // добавление меню "Файл" в строку меню
   MenuItem open = new MenuItem ("Открыть");// создание команды "Открыть"
   open.addActionListener(new java.awt.event.ActionListener() {    // обработчик события команды "Открыть"
     public void actionPerformed(ActionEvent e) {
     myApplet.info.setText("Вы выбрали Открыть");}
   });
   fileMenu.add(open);                           // добавление команды "Открыть" в меню "Файл"

// создание пункта меню, который можно отмечать флажком
  final CheckboxMenuItem autoSave = new CheckboxMenuItem ("Автосохранение");
   autoSave.addItemListener(new java.awt.event.ItemListener() { // обработчик события изменения флажка
   public void itemStateChanged(ItemEvent e) {
   if (autoSave.getState()){myApplet.info.setText("Автосохранение включено");}
   if (!autoSave.getState()){myApplet.info.setText("Автосохранение отключено");}
   }});
   fileMenu.add(autoSave);                       // добавление команды меню "Автосохранение"
   fileMenu.addSeparator();                      // добавление разделителя меню
   Menu subMenu = new Menu ("Подменю");          // создание подменю
   fileMenu.add(subMenu);                        // добавление подменю в меню "Файл"
   MenuItem Close = new MenuItem ("Закрыть"); // создание команды меню "Закрыть"
   subMenu.add(Close);                           // добавление команды "Закрыть" в подменю
   Close.disable();                              // отключение команды "Закрыть"
   Close.addActionListener(new java.awt.event.ActionListener( ) {  // обработка выбора команды  "Закрыть"
     public void actionPerformed(ActionEvent e) { myFrame.hide( ); }
   });

   myFrame.show();                               // отображение фрейма

   // создание диалога. true указывает на то, что диалог модальный
   final Dialog myDialog = new Dialog(myFrame, "Заголовок диалога", true);
   myDialog.setLayout(new FlowLayout());                    // менеджер расположения для диалога
   myDialog.add(new Label("Нажмите ОК для начала работы с апплетом ...")); // добавление надписи
Button OK = new Button("OK");                                                // создание кнопки OK
   OK.addMouseListener(new java.awt.event.MouseAdapter() { // обработчик события нажатия кнопки OK
   public void mouseClicked(MouseEvent e) {myDialog.hide();} });            // закрывает диалог
   myDialog.add(OK);                                                          // добавление кнопки OK в диалог
   myDialog.pack();                              // "ужимает" диалог до минимально необходимых размеров
   myDialog.setResizable(false);                              // запрет изменения размера окна диалога
   myDialog.show();                                           // отображение диалога
}}
```

10.5. Взаимодействие апплета с сервером (пакет java.net).

Язык Java обладает мощными возможностями организациями сетевого взаимодействия по модели клиент-сервер, большинство из которых входят в пакет java.net. Это, например, класс Soket, позволяющий организовать соединение по протоколу TCP между различными портами клиента и сервера, а также класс DatagramPacket, реализующий протокол UDP, класс MulticastSocket, позволяющий использовать широковещательные пакеты, и многие другие. К сожалению, из-за объемности вопроса, нет возможности изложить его в лекциях, так что желающим придется проработать его самостоятельно (при рассмотрении вопроса, надо не забывать, что апплету разрешено устанавливать сетевые соединения только с сервером откуда он был загружен). Ниже будет приведено только несколько примеров использования класса URL и

URLConnection, которые позволяют получать от сервера (передавать на сервер) информацию по протоколу HTTP.

Пример 1. Считывается html – документ по указанному адресу и отображается в апплете:

```
import java.applet.*;  import java.awt.*;  import java.net.*;
try {
URL address=new URL ("http://www.microsoft.com");        // создание объекта URL
AppletContext document = getAppletContext ( );            // получение ссылки на контекст апплета
document.showDocument (address);                          // отображение документа с www.microsoft.com
}
catch (MalformedURLException e) { /* не удалось прочитать */ }
```

Пример 2. На сервер, CGI-приложению, методом POST, посылается строка ПРОВЕРКА, и считывается ответ сервера – одна строка.

```
try {
URL address = new URL ("http://somewhere.com/cgi-bin/proga");      // создание объекта URL
URLConnection soedinenie = address.openConnection( ); /* более удобный объекта URLConnection обеспе-
                                                         чивает не посимвольный а построчный ввод */
PrintStream out = new PrintStream (soedinenie.getOutputStream ( ) );   // объекта out для записи данных
DataInputStream in = new DataInputStream (soedinenie.getInputStream ( ) ); // объект in для чтения данных
out.println("string= ПРОВЕРКА ");          // отправка строки на сервер
String otvet = in.readLine ( );            // чтение строки с сервера
}    catch (MalformedURLException e1) {}            catch (java.io.IOException e2) {} // обработка исключений
```

* Чтобы воспользоваться методом POST, достаточно записать данные в объект out прежде, чем выполнить чтение из объекта in. Если же сначала считать данные из input, то будет выполнен метод GET, и последующие попытки записи данных в output приведут к ошибке.

10.6. Параметры, конфигурирование апплета.

В апплетах могут использоваться параметры, задаваемые пользователем при запуске апплета. Это чем-то напоминает указание аргументов в командной строке, при запуске DOS программ. Параметры задаются при помощи тэга <PARAM>, находящегося внутри тэгов <APPLET> </APPLET>, определяющих апплет. В приведенном ниже примере апплету передаются параметры stroka (отображаемая на экране строка) и font_size (размер шрифта):

```
<html> <body> <Applet code="AppletName.class" width=350  height=200 name="AppletName">
<PARAM NAME=stroka VALUE="Отображаемая строка">
<PARAM NAME=font_size VALUE=72>
</Applet> </body></html>
```

Переданные апплету параметры можно прочитать из апплета при помощи метода GetParameter ( ). Пример:

```
import java.awt.*;
import java.Applet.*;
public class AppletName extends Applet {
String str;
        public void init ( ) {
                String s; str = getParameter("stroka");  \\ считывание параметра stroka
                s = getParameter("font_size"); \\ считывание параметра font_size
                int FontSize = Integer.parseInt(s); \\ преобразование параметра font_size в число типа int
                Font font = new Font("TimesRoman", Font.BOLD, FontSize);
                setFont(font); \\ установление параметров шрифта                    }
public void paint(Graphics g) {  g.drawstring (str, 100, 100);  }
}
```

Заключение:

В лекции были рассмотрены только базовые конструкции языка Java. Не рассмотрен пакет java.io, позволяющий организовать работу с файлами (апплетам запрещены чтение и запись на диск). Не рассмотрена работа с потоками – мощная возможность, позволяющая реализовать параллельное выполнение участков программы. Не рассмотрен криптографический интерфейс Java Security API, позволяющий легко реализовать надежные криптографические алгоритмы (шифрование данных). Не рассмотрено взаимодейст-вие с менеджером безопасности java. Не рассмотрена возможность динамической загрузки классов, сервлеты Java (апплеты, которые выполняются на сервере, интегрируясь с ним), взаимодействие Java с другими языками, интерфейс Java Beans API (создание межплатформенных подключаемых модулей, представляющих собой законченные объекты из которых можно "собирать" программы и пользовательский интерфейс), использование Java для доступа к базам данных и многое другое. Однако хочется надеяться, что

полученные базовые знания помогут вам легко читать Java-программы и самостоятельно разобраться в этих вопросах.

Лекция 12. Язык JavaScript

JavaScript - язык для составления скриптов, встраиваемых в Web-страницы для придания им привлекательного вида. В отличие от VBScript, который поддерживается только в Internet Explorer, JavaScript поддерживается большинством популярных браузеров. JavaScript - это не Java ! Язык Java был разработан компанией Sun Microsistems, а JavaScript – фирмой Netscape (первоначально этот продукт Netscape назывался LiveScript и только после бума Java, фирма Netscape купила у Sun лицензию и переименовало свое детище в JavaScript). Хотя синтаксис Java и JavaScript во многом схож, однако возможности и назначение языков разные. Если апплеты Java представляют собой независимые программы с широкой гаммой возможностей, встраиваемые в HTML-страницу, то скрипты JavaScript в основном нацелены на использование существующей объектной модели Web-браузера, а также позволяют расширять ее, создавая собственные объекты. Программы на Java представляют собой байт-код, а программы на JavaScript непосредственно включаются в Web-страницу в виде исходного текста. Таким образом, Java – это язык для написания программ, а JavaScript – это язык для написания скриптов управления HTML-страницами.

Для включения программ на JavaScript в HTML-страницу необходимы те же теги <SCRIPT> </SCRIPT>, что и для программ на VBScript, выглядящие примерно так:

```
<SCRIPT LANGUAGE="JavaScript">
<!--
alert("Приветик");
-->
</SCRIPT>
```

В остальном, принцип размещения и написания JavaScript программ похож на рассмотренный ранее для VBScript – тот же объектно-ориентированный подход, то же манипулирование объектами Window, Document, Form и др. через их свойства, методы и события. Единственное отличие – это синтаксис, унаследованный от Java. В лекциях не будет уделяться много внимания синтаксису JavaScript, за исключением случаев, когда он отличен от синтаксиса Java. Основной упор будет сделан на решение ряда задач (создание анимации, управление слоями и др.). Поэтому рекомендую перечитать лекции по Java, а также лекции по VBScript и HTML.

Существует четыре версии JavaScript.

- JavaScript 1.0. Первая версия языка, поддерживаемая Internet Explorer 3.0 и Netscape Navigator 2.0.
- JavaScript 1.1. Поддерживается Netscape Navigator 3.0 и Internet Explorer 4.0 (почти полностью).
- JavaScript 1.2. Поддерживается Netscape Navigator 4.0 и Internet Explorer 4.0 (частично).
- JavaScript 1.3. Поддерживается Netscape Navigator 4.5.

В дескрипторе <SCRIPT> можно указывать номер версии JavaScript. Указание версии JavaScript предотвращает попытки броузеров старых версий выполнить сценарии, созданные в новых версиях JavaScript. Если вы используете JavaScript 1.3, то Netscape 4.0, Internet Explorer 4.0 и ранних версий выполнить сценарии не в состоянии.

Начиная с версии JavaScript 1.2, программы на JavaScript могут не включаться непосредственно в текст HTML-страницы, а быть оформлены в виде отдельного файла, с расширением js. Пример:

```
<SCRIPT LANGUAGE="JavaScript1.2"  SRC="programma.js">
</SCRIPT>
```

Описание функций в JavaScript.

Программа на JavaScript может включаться в HTML – документ непосредственно между тэгами <SCRIPT> </SCRIPT>, как это было показано выше. В этом случае она будет выполнена после того, как браузер отобразит содержимое документа предшествующее тэгу <SCRIPT>, и до того, как браузер отобразит содержимое документа, следующее за тэгом </SCRIPT> (см. лекции по VBScript). Однако чаще всего, программы на JavaScript оформляют в виде отдельных функций, которые запускаются в ответ на определенные события (щелчок мышью, перемещение над гиперссылкой и т.д.). Функции оформляются следующим образом:

Синтаксис:

```
function ИмяФункции ( ) {
	команда;
	return  возвращаемое_значение;
}
```

Пример:
```
<HTML> <HEAD>
<SCRIPT LANGUAGE="JavaScript">
function Greet (who) {
alert("Приветик " + who); }
</SCRIPT> </HEAD>
<BODY>
Приветствие выводится два раза
<SCRIPT LANGUAGE="JavaScript">
Greet("Ваня"); Greet("Серега");
</SCRIPT>
</BODY>
</HTML>
```

Обработка событий в JavaScript.

Каждый объект HTML-страницы: кнопка, гиперссылка, сама страница и др. – имеют свой набор событий (см. лекцию по VBScript). Обработка событий (щелчки мышью, наведение на гиперссылку, загрузка страницы и т.д.) реализуется в JavaScript в виде отдельной функции, либо непосредственно в самом теге элемента. Можно также динамически изменять события назначенные элементу. Примеры задания обработчиков событий приведены ниже:

Пример 1. Обработчик события в тэге элемента:
```
<HTML><BODY>
<a href="http://www.microsoft.com" onmouseover="alert('Без стука не входить!'); ">  Гиперссылка </a>
</BODY> </HTML>
```

Пример 2. Обработчик события в виде отдельной функции:
```
<HTML>
<HEAD> <SCRIPT LANGUAGE="JavaScript">
function info( ) {alert(' Без стука не входить!'); }
</SCRIPT> </HEAD>
<BODY>
<a href="http://www.microsoft.com" onmouseover="info ();">  Гиперссылка </a>
</BODY> </HTML>
```

Пример 3. Динамическое назначение обработчика событий:
```
<HTML><HEAD>
<SCRIPT LANGUAGE="JavaScript">
function info ( ) { alert(' Без стука не входить!'); }
</SCRIPT> </HEAD>
<BODY>
<a href="http://www.microsoft.com" name="link1">  Гиперссылка </a>

<SCRIPT LANGUAGE="JavaScript"> link1.onmouseover=info; </SCRIPT>
</BODY> </HTML>
```

Типы данных, глобальные и локальные переменные в JavaScript.

В некоторых языках программирования необходимо при объявлении переменных указывать их тип данных. В JavaScript тип данных переменной не указывается. Если переменной, содержащей целое число присвоить строковое значение, то ошибки не произойдет, т.к. переменная имеет "любой" тип данных, который может включать в себя:

- Числовой тип (целые числа или числа с плавающей точкой).
- Булевский тип, или логический (true (истина) или false (ложь)).
- Строковый тип.
- Нулевой тип. Определяется ключевым словом null. Если переменная не была определена, то она принимает это значение.

В JavaScript переменные можно использовать без предварительного описания. Если же переменная все-таки описывается, то это делается следующим образом:

Синтаксис:
```
1) переменная = значение;
2) var переменная = значение;
```

Пример:
```
bird = "Альбатрос";
var bird = "Альбатрос";
```

Ключевое слово **var** дает указание JavaScript создать локальную переменную, даже если уже существует глобальная переменная с таким же именем. Хотя это и не обязательно, для избежания ошибок, всегда используйте var при объявлении локальных переменных.

Все переменные в JavaScript делятся на локальные (описанные на уровне функции) и глобальные (описанные вне какой-либо функции, но между тэгами <SCRIPT> </SCRIPT>). Локальные переменные видны только в рамках одной функции и существуют только пока выполняется функция. Глобальные переменные, описанные в рамках какого-либо тэга <SCRIPT> </SCRIPT>, видны для всех функций из любых тэгов <SCRIPT> </SCRIPT> на данной странице, и существуют, пока загружена страница.

Пример:

```
<HTML> <HEAD>
<SCRIPT LANGUAGE="JavaScript">
name1="Ваня";   name2="Серега";
function Greet(who) {
var name3="Таня";   alert("Внимание!" + who);
}
</SCRIPT>
</HEAD>
</BODY>  Сообщение выводится два раза. Переменная name3 недоступна, т.к. она локальная.
<SCRIPT LANGUAGE="JavaScript">  Greet(namel);  Greet(name2);  </SCRIPT>
</BODY> </HTML>
```

Преобразование типов данных

JavaScript во всех возможных случаях самостоятельно производит преобразование одних типов данных в другие. Пусть вы в программе используете оператор: document.write("Общая сумма: " + summa);
Если переменная summa имеет значение 40, то на экране отобразится следующая строка: Общая сумма: 40.
Любые нетекстовые значения (в нашем примере summa) преобразуются в текстовые. И только после этого результат выводится в окне браузера. В некоторых ситуациях сроковый тип данных необходимо преобразовать в числовой. Для этих целей в JavaScript используются две функции.

- parselnt(). Преобразует текстовый тип данных в целочисленный.
- parseFloat(). Преобразует текстовый тип данных в числовой с плавающей точкой.

Пример:

```
stroka = "2000 ершиков";
chislo  = parseInt(stroka);
```

После выполнения этих операторов переменная chislo принимает значение 2000. Нечисловая часть предложения игнорируется и отбрасывается. Функции преобразования типов данных ищут только первое число в строке текста. Если число не найдено, функция возвращает строковое значение NaN, указывая на то, что текст не содержит числовых значений.

Массивы в JavaScript.

Массивы необходимо объявлять перед использованием. В приведенном ниже примере объявлен массив, состоящий из 30 элементов: massiv = new Array(30);
Индексация в массиве начинается с нуля. Следующие операторы определяют значения первых трех элементов массива: massiv[0] = 39; massiv[1] =40; massiv[2] = 100;

Подобно строковым переменным массивы имеют свойство length. Оно определяет количество элементов, из которых состоит массив. Пример: dlina_massiva = massiv.length ;

Любой массив имеет метод sort( ), используемый для сортировки элементов массива. Он возвращает упорядоченную копию исходного массива. Упорядочение проводится как по алфавиту (для строковых значений), так и по возрастанию или убыванию (для числовых значений). Пример:

```
massiv = new Array(3);
massiv [ 0 ] = "Сидоров";   massiv [ 1 ] = "Иванов";    massiv [ 2 ] = "Петров";
sorted_massiv = massiv.sort ( );
```

Для массивов и объектов существует специальный цикл for.. .in, который последовательно перебирает каждый элемент массива (объект).

Пример 1 (массив):

```
for (i in massiv)  { document.writeln ( massiv [ i ] ); }
```

Пример 2 (объект):

```
for (i in navigator) { document.write(" Значение: "+navigator[i]); }
```

Арифметические и логические операции.

<table>
<tr><th>Операция</th><th>Описание</th></tr>
<tr><td>x = 3</td><td>Присвоить переменной x значение 3.</td></tr>
<tr><td>+, -, *, /, %</td><td>Сложение, вычитание, умножение, деление и деление по модулю (определение остатка от деления).</td></tr>
<tr><td>x ++, x --</td><td>Увеличить x на единицу, уменьшить x на единицу.</td></tr>
<tr><td>+=, -=, *=, /=</td><td>Сложение, вычитание, умножение и деление с присваиванием.
x+=3 аналогично команде x=x+3.</td></tr>
<tr><td>x == 3</td><td>Операция равенства: x равен 3. Используется в конструкции IF..ELSE и др.
Пример: if (x==3)   { //действия };</td></tr>
<tr><td>x != 3</td><td>Операция неравенства: x не равен 3.</td></tr>
<tr><td><,   <=,   >,   >=</td><td>Операции отношений: меньше, меньше или равно, больше, больше или равно.</td></tr>
<tr><td>x | y</td><td>Поразрядная операция ИЛИ (OR). Результатом является число, полученное в соответствии с таблицей истинности для ИЛИ. Пример:
x: 00001010   =   10
y: 00001100   =   12
Результат: 00001110   =   14</td></tr>
<tr><td>x & y</td><td>Поразрядная операция И (AND).</td></tr>
<tr><td>x ^ y</td><td>Поразрядная операция исключающее ИЛИ (XOR).</td></tr>
<tr><td>~ x</td><td>Поразрядное логическое отрицание НЕ (NOT).</td></tr>
<tr><td>&=, |=,   ^=</td><td>Поразрядные операции с присваиванием: x&=y аналогично x=x&y</td></tr>
<tr><td>x | | y</td><td>Логическая операция ИЛИ (OR). В отличие от поразрядных операций, результат логической операции не число, а только значение ИСТИНА или ЛОЖЬ. Именно логические операции используются в конструкциях IF..ELSE и др.</td></tr>
<tr><td>x && y</td><td>Логическая операция И (AND).</td></tr>
<tr><td>!x</td><td>Логическая операция НЕ (NOT).</td></tr>
<tr><td>-x</td><td>Унарная операция "изменение знака".</td></tr>
<tr><td>x<<2</td><td>Поразрядный сдвиг битов влево: сдвинуть биты переменной x на 2 бита влево. Пример: 00000001 (двоичное) <<2 дает 00000100 (двоичное).</td></tr>
<tr><td>>>, >>></td><td>Поразрядный сдвиг вправо, поразрядный сдвиг вправо без знака.</td></tr>
<tr><td><<=, >>=, >>>=</td><td>Поразрядный сдвиг с присваиванием.</td></tr>
</table>

Циклы и управляющие операторы.

Оператор if - else:

Синтаксис: if   (выражение) {операторы;}   else   {операторы; }
Пример: 1) if (x==2 && x!=3) { /*операторы выполняются если x равен 2 и не равен 3*/ } else { /*операторы */ } 2) if (x==2) { /*операторы */ }

Оператор while

Синтаксис: 1) while (выражение) {операторы;} 2) do {операторы;} while (выражение);
Пример: 1) while(x==2 && x!=3) { x++; /* и др. операторы - выполняются пока x равен 2 и не равен 3 */ } 2) do { /*операторы */ } while (x==2);

Операторы for

Синтаксис: 1) for (счетчик=исходное_значение; (условие_остановки); шаг) { /*операторы */ }
Пример: 1) for (i=0, j=10;   ((i >100) | | yslovie_J ( ));   i++, raschet_J ( )) { /*операторы */ }

2) for (i=0; i==100; i++) {/*операторы */ }

Операторы switch

```
Синтаксис:
switch (переключатель) {
        case значение1 : оператор1; операторN; break;
        case значениеN: оператор1; операторN; break;
         default:  оператор1; операторN; break;
}
```

```
Пример:
<HTML> <BODY>  <SCRIPT LANGUAGE="JavaScript">
where = window.prompt("Куда заглянем сегодня?");
switch (where) {
case "Netscape" :  window.location="http://www.netscape.com"; break;
case "Microsoft" :  window.location="http://www.microsoft.com";  break;
case "Yahoo" :  window. location=" http: / /www. yahoo. com";  break;
default :  window.location="http://www.mcp.com"; }
</SCRIPT> </BODY> </HTML>
```

Математические функции, дата и время, работа со строками

Математические функции:

Класс Math, который предоставляет некоторые полезные математические и тригонометрические функции.

Пример:

```
chislo1 = Math.min (100, 700); // Возвращает минимальное значения из двух чисел 100 и 700
chislo2 = Math.max (100, 700); // Возвращает максимальное значения из двух чисел 100 и 700
chislo3 = Math.abs (-3); // Возвращает абсолютное значение числа
chislo4 = 10*Math.random ( ) + 1;  // Возвращает случайное число от 1 до 10.
chislo5 =  Math.round (10.5);  //Округление по правилам математики. Результат = 11
chislo6 =  Math.floor(10.7);  //Округление всегда в меньшую сторону. Результат = 10
chislo7 =  Math.ceil (10.3);  //Округление всегда в большую сторону. Результат = 11
chislo8 =  Math.sqrt ( 4 ); //Квадратный корень из 4.
chislo9 =  Math.pow ( 7, 1/3 ); // Число 7 в степени 1/3
chislo10 =  Math.log ( 7 ); // Натуральный логарифм числа 7.
chislo11 =  Math.log ( 7 ) / Math.log ( 10 ); // Десятичный логарифм числа 7.
chislo12 =  Math.exp ( 7 ); // Экспонента числа 7, т.е. e^7.  Существует константа Math.E
chislo13 =  Math.sin ( 3.14 ); // Синус. Угол задается в радианах. Радианы=градусы*Math.PI / 180.
chislo14 =  Math.cos ( Math.PI / 2 ); // Косинус. Угол задается в радианах.
chislo15 =  Math.tan ( 3.14 ); /* Тангенс. Существуют также методы asin(), acos(), atan(), означающие
                    арккосинус, арксинус и арктангенс, соответственно. */
```

Дата и время:

Для работы со значениями даты и времени можно воспользоваться следующими конструкциями:

```
timer2 = new Date ();               // Создание объекта "дата"
vremya1 = timer2.getYear();         // Получить число лет, прошедших с 1900 г, т.е. для 2001 = 101
vremya2 = timer2.getMonth();        // Получить текущий месяц. Январь = 0, Февраль = 1 и т.д.
vremya3 = timer2.getDate();         // Получить текущую дату
vremya4 = timer2.getHours();        // Получить часы
vremya5 = timer2.getMinutes();      // Получить минуты
vremya6 = timer2.getSeconds();      // Получить секунды
vremya7 = timer2.getDay();          // Получить день недели: 0-воскресенье, 1-понедельник и т.д.
timer2.setYear(102); // Установить 2002 год
timer2.setMonth(11); // Установить 12-й месяц
timer2.setDate(20);
timer2.setHours(23);
timer2.setMinutes(5);
timer2.setSeconds(10);
```

Работа со строками:

Над строками можно выполнять некоторые функции, в результате которых создаются новые измененные строки (примеры см. ниже).

```
komy = "вам" ; // или komy = new String("вам");
summa = "Привет "+komy+"братья" ;
```

Переносить часть строки на новую строчку нельзя. В тексте строки нельзя использовать двойные или одинарные кавычки и косую черту "\". Если же необходимо этими символами все-таки воспользоваться, то применяют управляющие текстовые и восьмеричные константы:

Константа	Значение
\n	перевод строки
\f	перевод формата
\r	возврат каретки
\"	\u0022 двойная кавычка
\'	\u0027 одиночная кавычка
\\	\u005c обратная косая черта
\007	\u0007 звонок
\101	\u0041 буква"А"
\071	\u0039 цифра "9"

Пример:
primer = "На разных \n строках и \" в кавычках \" буква \110, со звонком \007"

Ниже приведены примеры некоторых функций для работы со строками:

```
stroka = “Это строка”;  // создание строки stroka
otvet = stroka.startsWith(‘Это’); /* переменная otvet будет true (истина), если строка начинается со слова
                                          “Это”, регистр учитывается */
otvet = stroka.endsWith(‘строка); /* переменная ответ будет true (истина), если строка кончается  словом
                                          “строка”, регистр учитывается */
otvet = stroka.indexOf('o');  // переменная otvet будет содержать номер позиции первой буквы "о" в строке
otvet = stroka.indexOf('o',otvet+1);  /* после того, как в предыдущем примере будет найдена первая буква "о",
                                        в этом примере поиск следующей буквы "о" в строке будет
                                        продолжен, начиная со следующей после "о" позиции. Так можно
                                        найти все буквы "о", продолжая поиск до тех пор, пока otvet != 0*/
otvet = stroka.lastIndexOf('o');  // тоже самое, что и indexOf('o'), но поиск начинается с конца строки
otvet = stroka.lastIndexOf('o', otvet-1 );  // аналогично примеру выше
otvet = stroka.indexOf('Это');  /* можно определять позицию не только отдельного символа, но и подстроки */
bukva = stroka.charAt (3); /* в переменную bukva будет помещен 4-й символ строки (индекс первого символа
                               строки = 0) */
stroka2 = stroka.subString(4 ); /* выделение подстроки из строки "stroka", начиная с 5-ой позиции и до конца
                                     строки */
stroka2 = stroka.subString(4, 9 ); /* выделение подстроки из строки "stroka", начиная с 5-ой и заканчивая 10-ой
                                     позицией строки */
stroka2 = stroka.replace('o', 'a'); // заменяет в строке все буквы "о" на букву "а"
stroka2 = stroka.toUpperCase( ); // преобразует строку в верхний регистр
stroka2 = stroka.toLowerCase( ); // преобразует строку в нижний регистр
chislo=10;  stroka = String.valueOf(chislo); // переводит практически любой тип данных в тип String
```

Использование объекта event в обработчике события

Объект event используется в JavaScript версии 1.2 и выше. Это специальный объект, который отправляется в обработчик событий при возникновении любого события. Обработчик события получает этот объект в виде параметра. Свойства объекта event содержат данные о событии, которое произошло. Ниже приведен список всех свойств объекта event (для Microsoft Internet Explorer):

- type. Это тип произошедшего события, например mouseover.
- keyCode. Код нажатой пользователем клавиши.
- altKey. Принимает значение true, если удерживается клавиша "alt" и false в противном случае.
- ctrlKey. Принимает значение true, если удерживается клавиша "ctrl" и false в противном случае.
- shiftKey. Принимает значение true, если удерживается клавиша "shift" и false в противном случае.
- button. Код нажатой кнопки мыши.
- X и Y. Это координаты указателя мыши вдоль оси X и Y. Начало координат находится в левом верхнем углу окна Web-браузера.
- screenX и screenY. Это координаты указателя мыши вдоль оси X и Y. Начало координат находится в левом верхнем углу экрана.

Пример использованиz объекта Event для определения нажатой клавиши:

```
<HTML> <BODY onkeypress="window.alert('Вы нажали клавишу:   '+String.fromCharCode(event.keyCode));">
</BODY> </HTML>
```

Создание пользовательских объектов

В JavaScript можно использовать встроенные объекты Web-браузера, аналогично тому, как это описывалось в лекции поVBScript, а можно создавать и собственные объекты и даже добавлять новые свойства и методы во встроенные объекты Web-браузера.

Создание собственного объекта происходит следующим образом:

1) Задается функция-конструктор объекта, инициализирующий его свойства и методы. Пример:

```
function Cartochka ( name, address, telefon )  {
this.name = name; this.address = address; this.telefon = telefon;
this . printMetod = printMetod;                              }
```

2) Описываются методы объекта, заданные в конструкторе. Пример:

```
function printMetod ( ) {
i=this.name+"<BR>"+this.adres+"<BR>"+this.telefon+"<BR>" ;
document.writeln ( i ) ; }
```

3) Создание экземпляра объекта. Пример:

```
Ivan = new Cartochka ("Иван", "Мелитополь","555-00-00");
```

4) Использование свойств и методов объекта. Пример:

```
Ivan.telefon = "03";  Ivan.printMetod();
```

Полный текст соответствующего HTML-документа приведен ниже:

```
<HTML> <BODY> <SCRIPT LANGUAGE="JavaScript">
function printMetod ( )  {
i=this.name+"<BR>"+this.adres+"<BR>"+this.telefon+"<BR>"
document.writeln(i)       }

function Cartochka ( name, adres, telefon )  {
this.name = name; this.adres = adres; this.telefon = telefon;
this.printMetod = printMetod;                    }

Ivan = new Cartochka ("Иван", "Мелитополь","555-00-00");
Ivan.telefon = "03";  Ivan.printMetod();
</SCRIPT> </BODY> </HTML>
```

Создание иерархии подчиненных объектов реализуется аналогично Java: в конструкторе объекта верхнего уровня записывается свойство, которое создает и указывает на объект нижнего уровня. Таким образом, создавая объект верхнего уровня, автоматически создается вся иерархия подчиненных объектов. В приведенном ниже примере, объект верхнего уровня kniga1, содержит массив подчиненных объектов Cartochka:

```
function Kniga ( )  {
this . stranica = new Array(2);
this . stranica[0] =  new Cartochka ("Иван", "Мелитополь","555-00-00");
this . stranica[1] =  new Cartochka ("Петр", "Мелитополь","555-00-01");
this . stranica[2] =  new Cartochka ("Игорь", "Мелитополь","555-00-02");
}
kniga1 = new Kniga ( );
kniga1. stranica[1].name = "Иванов";
kniga1. stranica[1].printMetod();
```

Настройка встроенных объектов Web-браузера

Добавление свойств и методов в уже существующий встроенный объект проводится с помощью ключевого слова prototype (другими словами, создается *прототип* уже существующего объекта. Prototype — это, в данном случае, название конструктора объекта.). Последовательность действий следующая:

1) Описать функцию (переменную), которая станет новым методом (свойством) встроенного объекта.
   Пример:

```
function Metod (  )  {  document.writeln (" Этот метод добавлен ко встроенному объекту ! "); }
Svoistvo = " Это новое свойство встроенного объекта ";
```

2) Добавить метод (свойство) во встроенный объект, используя ключевое слово prototype. Пример:

```
String.prototype.newMetod = Metod;
String.prototype.newSvoistvo = Svoistvo;
```

3) Использование методов и свойств. Пример:

```
stroka = new String ("Строка");
stroka . newMetod ( );   document.writeln ( stroka . newSvoistvo );
```

Отображение бегущих строк

Используя функции работы со строками и объект Math, описанные в лекциях по Java, а также метод SetTimeout объекта Window (см. лекции по VBScript) можно создать бегущие строки. Пример приведен ниже:

```
<HTML> <HEAD>
<SCRIPT LANGUAGE="JavaScript">
var msg = "Это пример бегущей строки.    Впечетляет?";
spacer = "...           ...";
pos = 0;
function ScrollMessage(){
window.status = msg.substring(pos, msg.length) + spacer + msg.substring(0,pos);
pos++;
if (pos > msg.length) pos = 0;
window.setTimeout("ScrollMessage()",100); }
ScrollMessage();
</SCRIPT> </HEAD>
<BODY> <H1> Пример бегущей строки</H1> </BODY>
</HTML>
```

Создание гиперссылки в виде рисунка, меняющегося при наведении на него указателя мыши

Для создания такого рисунка достаточно поместить его внутрь тэга гиперссылки, и написать в тэге <IMG>, задающем рисунок, обработчики событий onmouseover и onmouseout, подменяющие и восстанавливающие рисунок. Пример:

```
<HTML> <BODY>
<A HREF="http://microsoft.com">
<IMG src="file1.gif" name="ris1" width="100px" height="100px" alt="ссылка " border="0"
onmouseover="ris1 . src = 'file2.gif ';"  onmouseout="ris1 . src = 'file1.gif ';"> </A>
</BODY> </HTML>
```

Создание анимации с помощью массивов

Простейшая анимация реализуется путем создания анимированного gif рисунка в графических редакторах. Однако анимацию можно выполнить и при помощи JavaScript. Рисунок, внедренный в HTML - страницу, представляют собой объект image, дочерний по отношению к объекту document. Каждый объект image имеет следующие свойства:

- border. Соответствует атрибуту BORDER дескриптора <IMG>. Определяет границы рисунка.
- complete.Определяет степень загруженности рисунка. Принимает булевы значения (true или false).
- heigth и width. Задают размеры рисунка. Свойства только для чтения. Изменить их при создании динамических рисунков нельзя.
- hspace и vspace. Определяют место расположения рисунка на странице. Только для чтения.
- name. Имя рисунка. Оно определяется атрибутом NAME при определении рисунка.
- lowscr. Принимает значение атрибута LOWSCR. Это специальный атрибут, используемый броузером, который определяет загрузку рисунка в низком разрешении перед загрузкой основного изображения.
- src. Источник рисунка, определяемый адресом URL. Это свойство может изменяться.

При создания анимации, исходный рисунок на странице подменяется рисунками из массива. Пример:

```
<HTML> <HEAD> Пример анимации </HEAD>
<BODY>
<IMG border="0" src="1.jpg" alt="анимация">
<SCRIPT LANGUAGE="JavaScript">
var ind=1;          massiv=new Array(3);
image1=new Image();      image1.src="1.jpg";
image2=new Image();      image2.src="2.jpg";
image3=new Image();      image3.src="3.jpg";
massiv[1]=image1;         massiv[2]=image2;          massiv[3]=image3;

function ScrollPicture(){
document.images[0].src=massiv[ind].src;
ind=ind+1;          if (ind>3) {ind=1}
window.setTimeout("ScrollPicture()",500); }
```

```
ScrollPicture();
</SCRIPT>
</BODY>
</HTML>
```

Создание динамических страниц с помощью слоев (DHTML)

Содержание документа HTML может находится на различных слоях (подробнее см. лекцию по HTML, раздел слои DHTML и стили CSS). Слои могут быть наложены друг на друга в определенном порядке и перекрываться. С помощью JavaScript их также можно перемещать, скрывать и отображать. Каждый слой имеет свои уникальные свойства, например цвет или рисунок фона.

Определить слой можно несколькими способами, но самый популярный из них заключается в использовании дескриптора <DIV>, который впервые стали использовать в HTML 3.0. Чтобы создать слой с помощью дескриптора <DIV>, заключите содержимое слоя в пару дескрипторов <DIV> и определите свойства после атрибута STYLE. Пример:

```
<DIV ID="nazvanie1" STYLE="position:absolute; left:100; top=100">
Это содержимое слоя
</DIV>
```

Этот код определяет слой nazvanie1. Это перемещаемый слой, смешенный относительно левого верхнего угла окна браузера на 100 по вертикали и 100 пикселей по горизонтали.

В атрибуте STYLE дескриптора <DIV> можно указывать самые различные свойства слоев. Доступ к этим свойствам возможен и из программ на JavaScript. Некоторые свойства слоя приведены ниже:

- position. Определяет расположение слоя в окне. Принимает три значения: static (неперемещаемый слой), absolute (координаты слоя – абсолютные, в пикселях относительно окна браузера), relative (координаты слоя – относительные, относительно его нормального расположения).
- left и top. Определяют координаты слоя: сдвиг слева и сверху. Для значения absolute отсчет ведется относительно окна браузера, а для relative — относительно нормального расположения слоя.
- width и height. Ширина и высота слоя.
- clip. Определяет границу на слое. Отображается только та часть слоя, которая расположена внутри границы.
- overflow. Режим усечение границей слоя. Принимает значения none (нет усечения), clip (есть усечение) и scroll (если слой не помещается в границу, то используются полосы прокрутки).
- zIndex. Определяет порядок наслоения слоев. Самый нижний слой имеет значение этого индекса 1. Слой, расположенный над ним, имеет индекс 2 и т.д.
- visibility. Определяет видимость слоя. Принимает значения visible (слой отображается), hidden (слой невидим) и inherit (если два слоя вложены друг в друга, то если отображается основной слой, то отображается и вложенный).
- backgroundColor. Цвет фона слоя.
- color. Цвет шрифта. Например "red".
- fontSize. Размер шрифта в пикселях.
- fontFamily. Название шрифта, например "Times".
- fontStyle. Тип шрифта (жирный, наклонный). Например "Italic".
- textAlign. Выравнивание текста. Возможны варианты "Left", "Right", "Center".
- letterSpacing. Расстояние между символами. Например 3.
- lineHeight. Высота строки.

JavaScript позволяет обращаться к слоям следующим образом:

Для Internet Explorer	Для Netscape Navigator	Для Netscape Navigator 6.0
document.all.имя_слоя.style.свойсво_слоя = значение; * имя_слоя определяется в параметре ID, тэга DIV	document.имя_слоя.свойство_слоя = значение;	document. getElementById(" имя_слоя").style.свойство_слоя = значение;

Ниже, в качестве примера, приведена анимация, реализованная при помощи слоев.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML> <HEAD> <TITLE>Создание анимации с помощью DHTML</TITLE>
<META content="text/html; charset=windows-1251" http-equiv="Content-Type">
<SCRIPT LANGUAGE="JavaScript">

/* определение типа браузера. Переменные: ns (Netscape),  ns6 (Netscape 6.0), ie (MS Internet Explorer).*/
ns=(document.layers)?1:0;
ns6=(document.getElementById&&!document.all)?1:0;
```

```
ie=(document.all)?1:0;


var pos1x=0; var pos1y=0;  var pos2x=550; var pos2y=0;
var speed1=Math.floor(Math.random()*10)+5;  var speed2=Math.floor(Math.random()*10)+5;
function next(){
pos1x=pos1x+speed1;      pos1y=pos1y+speed1;
if (pos1x>550) pos1x=0;  if (pos1y>300)  pos1y=0;

pos2x=pos2x-speed2;      pos2y=pos2y+speed2;
if (pos2x<0) pos2x=550;  if (pos2y>300) pos2y=0;

if (ns) {
document.ris1.left=poslx; document.ris1.top=pos1y;
document.ris2.left=pos2x; document.ris2.top=pos2y; }

if (ie) {
document.all.ris1.style.left=pos1x; document.all.ris1.style.top=pos1y;
document.all.ris2.style.left=pos2x; document.all.ris2.style.top=pos2y; }

if (ns6) {
document.getElementById("ris1").style.left=pos1x; document.getElementById("ris1").style.top=pos1y;
document.getElementById("ris2").style.left=pos2x; document.getElementById("ris2").style.top=pos2y; }

window.setTimeout("next();", 10);
}
</SCRIPT>
</HEAD>
<BODY onLoad="next();">
<H1>Создание анимации с помощью DHTML</H1> <HR>
<DIV ID="ris1" STYLE="position:absolute; left:0; top:0; width:200; height:100; visibility:show">
<A HREF="http:\\www.microsoft.com">Просто ссылка</A>
</DIV>
<DIV ID="ris2" STYLE="position:absolute; left:550; top:0; width:200; height:100; visibility:show">
<IMG src="kartinka2.gif" width="100px" height="100px" alt="картинка" border="0">
</DIV>
</BODY>
</HTML>
```

Зависимость программ на JavaScript от типа браузера

Как видно из кода примера предыдущего раздела, программы JavaScript, выполняющиеся в различных браузерах, отличаются, т.к. отличаются объектные модели самих браузеров. Поэтому первым шагом программы должно быть определение типа браузера. После чего, в необходимых точках программы предусматривается выполнение различных блоков команд для различных типов браузеров.

Для определения типа браузера могут использоваться различные приемы. В частности, можно пользоваться свойствами navigator.appName (название браузера) и navigator.appVersion (версия браузера). Однако мне более удобным и элегантным кажется метод, основанный на различной реализации браузерами объектов для работы со слоями.

Пример, приведенный ниже, иллюстрирует этот прием. В нем определяется тип браузера и, в зависимости от этого, используется разный программный код для организации анимации рисунка, находящегося внутри слоя. В примере вводятся три переменные: ns (Netscape Navigator), ie (Internet Explorer) и ns6 (Netscape Navigator 6.0). Если, в результате проверки существования того или иного объекта, какая-либо переменная получает значение "истина", то значит мы имеем дело с соответствующим браузером. Такой подход выгоден и потому, что может существовать множество браузеров, название которых трудно перечислить, но которые копируют объектную модель соответствующих популярных браузеров. Так, например, работа со слоями в браузере Opera реализована аналогично Internet Explorer (document.all.название_слоя. style.свойства_слоя), а значит программы для Internet Explorer сработают и в браузере Opera.

Пример:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title> Определение типа браузера</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

```
</head><body>


<div id="elogo" style="position:absolute; top:10; left:10; width:100; height:82">
<img src="enot1.gif" alt="gilermo" width="100" height="82" name="enot">
</div>

/* определение типа браузера. */
<script language="JavaScript">
ns=(document.layers)?1:0;
ns6=(document.getElementById&&!document.all)?1:0;
ie=(document.all)?1:0;

/* предварительная загрузка рисунков для анимации */
me=new Array (4);
e1=new Image(); e2=new Image(); e3=new Image(); e4=new Image();
e1.src="enot1.gif"; e2.src="enot1m.gif"; e3.src="enot2.gif"; e4.src="enot2m.gif";
me[1]=e1; me[2]=e2; me[3]=e3; me[4]=e4; ienot=1;

/* анимация */
function goEnot() {
if (ns) document.elogo.document.enot.src=me[ienot].src;  // версия кода для Netscape Navigator
if (ns6||ie) document.enot.src=me[ienot].src;            // версия кода для Internet Explorer, Netscape 6, Opera
ienot++; if (ienot>2) ienot=1;
T=setTimeout("goEnot()",300);
}
goEnot();

</script>
</body>
</html>
```